2011 IEEE 52nd Annual Symposium on Foundations of Computer Science

Approximation Algorithms for Correlated Knapsacks and Non-Martingale Bandits

Anupam Gupta* Ravishankar Krishnaswamy* Marco Molinaro† R. Ravi†

Abstract— In the stochastic knapsack problem, we are given a knapsack of size B, and a set of items whose sizes and rewards are drawn from a known probability distribution. To know the actual size and reward we have to schedule the item—when it completes, we get to know these values. The goal is to schedule the items (possibly making adaptive decisions based on the sizes seen so far) to maximize the expected total reward of items which successfully pack into the knapsack. We know constant-factor approximations when (i) the rewards and sizes are independent, and (ii) we cannot prematurely cancel items after we schedule them. What if either or both assumptions are relaxed?

Related stochastic packing problems are the multi-armed bandit (and budgeted learning) problems; here one is given several arms which evolve in a specified stochastic fashion with each pull, and the goal is to (adaptively) decide which arms to pull, in order to maximize the expected reward obtained after B pulls in total. Much recent work on this problem focuses on the case when the evolution of each arm follows a martingale, i.e., when the expected reward from one pull of an arm is the same as the reward at the current state. What if the rewards do not form a martingale?

In this paper, we give $O(1)$-approximation algorithms for the stochastic knapsack problem with correlations and/or cancellations. Extending the ideas developed here, we give $O(1)$-approximations for MAB problems without the martingale assumption. Indeed, we can show that previously proposed linear programming relaxations for these problems have large integrality gaps. So we propose new time-indexed LP relaxations; using a decomposition and "gap-filling" approach, we convert these fractional solutions to distributions over strategies, and then use the LP values and the time ordering information from these strategies to devise randomized *adaptive* scheduling algorithms.

1. Introduction

Stochastic packing problems seem to be conceptually harder than their deterministic counterparts. For example, even though the deterministic single-knapsack problem is very well understood, this problem already becomes challenging in the stochastic setting. Arising in diverse situations [23], [7], these problems were first studied from an approximations perspective in an important paper of Dean et al. [10] (see also [9], [8]). They considered the *stochastic knapsack problem*, where each item has a random size and a random reward, and the sizes are revealed only after an item is placed into the knapsack; the goal is to give an adaptive strategy for picking items irrevocably in order to maximize the expected value of those fitting into a knapsack with size B. Via an LP relaxation and a rounding algorithm, they gave *non-adaptive* solutions with expected rewards that are (surprisingly) within a constant-factor of the best *adaptive* ones, resulting in a constant adaptivity gap (also a notion they introduced). However, the results required that (a) the random rewards and sizes are independent of each other, and (b) once an item was placed, it can not be prematurely canceled—it is easy to see that these assumptions change the nature of the problem significantly.

Another fundamental class of stochastic decision problems is the so-called *(finite horizon) multi-armed bandit problems*. Here we have a collection of k arms, each evolving according to a different (but known) Markov chain that makes a random state transition when the arm is pulled. The goal is to devise a strategy that pulls the different arms at most B times to maximize the expected final reward; here there are payoffs associated with the states and the final reward is some function of the payoffs at the states visited by the random transitions. For example, the reward function could be the sum of the payoffs of all visited states, or the maximum payoff over all the final states the arms are in (which captures the exploration-exploitation trade-off), etc. Variations of this problem were introduced as early as in the 1950s [26], with numerous applications in statistical estimation, resource allocation, etc. [13]—however, most of the literature only address infinite horizon models.

The above developments in the stochastic knapsack problem have been crucial for understanding the multi-armed bandit problem (and other budgeted learning problems) from an approximation perspective. Indeed, recent works [17], [16], [18], [15], [19] have used ideas from [10] to obtain the first multiplicative guarantees for several variants of the multi-armed bandit problems, but they all crucially rely on an additional assumption on the input, namely *the "martingale" property*: if an arm is some state u, one pull of this arm would bring an expected payoff equal to the payoff of state u itself.

The motivation for such an assumption comes from the application of MABs in Bayesian learning: each arm i is associated with an unknown distribution $\mathcal{D}_i$; the states of its Markov chain correspond to different priors for $\mathcal{D}_i$ based on previous observations. We start with some known prior for $\mathcal{D}_i$ (the initial state of the Markov chain) and every pull gives a sample from $\mathcal{D}_i$, which is used to update our prior (i.e., causes a state transition)—the payoff of a state is the expected reward according to the conditional distribution. Under certain assumptions on the distributions and update rules, these Markov chains would satisfy the martingale

*Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213. Supported by NSF grants CCF-0964474 and CCF-1016799. RK supported in part by an IBM Graduate Fellowship.

†Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA 15213. Supported by NSF grants CCF-0728841 and CCF-1143998.

0272-5428/11 $26.00 © 2011 IEEE
DOI 10.1109/FOCS.2011.48

property (see for instance [15]).

However, the martingale assumption does not hold in many interesting situations—the correlated stochastic knapsack being one such example. Perhaps more importantly, it is too restrictive when we use the arms to model objects which can *react* to our actions. Such examples appear in project allocation or marketing problems [2]: For example, arms may model costumers that require repeated "pulls" (marketing actions) before they transition to a high payoff state, while the intermediate states yield no payoff. In another example, arms could model advertisers competing for B ad impressions; pulling an arm corresponds to assigning an impression—however, each advertiser has a budget on the number of impression it is willing to pay for and the rewards are typically non-increasing (a similar model is considered in [5]), implying that such instances would violate the martingale property. On a technical side, obtaining guarantees without the martingale assumption requires new tools that depart from the current stochastic knapsack methods.

Our Results: We give a constant-factor polynomial-time approximation algorithm for the general version of the stochastic knapsack problem where rewards may be correlated with the sizes. Our techniques are general and also apply to the setting when items could be canceled prematurely. We then extend those ideas to give constant-factor approximation algorithms for MAB problems with Markovian transitions (for the sum and the max objectives), where the martingale property is not assumed.

1.1. Why Previous Ideas Don't Extend, and Our Techniques

One reason why stochastic packing problems are more difficult than deterministic ones is that, unlike in the deterministic setting, we cannot simply take a solution with expected reward R^* that packs into a knapsack of size $2B$ and get one with reward $\Omega(R^*)$ whilst fitting within the budget of B (by appropriately sub-selecting some items). In fact, in stochastic settings, there are examples where a budget of $2B$ can fetch much more reward than what a budget of size B can (Appendix B). Moreover, allowing premature cancellations can also drastically increase the solution value (Appendix A). The assumptions in previous works on stochastic knapsack and MAB avoided both issues, but we now need to tackle them.

Stochastic Knapsack: Dean et al. [10], [8] assume that the reward of an item is independent of its size, and also do not consider the possibility of canceling items prematurely. These assumptions simplify the structure of the optimal (adaptive) decision tree and make it possible to formulate a knapsack-style LP which captures Opt, and subsequently round it. However their LP relaxation performs poorly when either correlation or cancellation is allowed (Appendix A).

Multi-Armed Bandits: Obtaining approximations for MAB problems is a more complicated task, since cancellations are inherent in the problem formulation (i.e., any strategy may stop playing a particular arm and switch to another) and the payoff of an arm is naturally correlated with its current state. While the first issue is tackled by using more elaborate LPs with a flow-like structure that compute a probability distribution over the different times at which the LP stops playing an arm (e.g., [16]), the latter issue is less understood. Indeed, several papers on this topic present strategies that fetch an expected reward which is a constant-factor of an optimal solution's reward, but which may violate the budget by a constant factor. In order to obtain a good solution without violating the budget, they critically make use of the martingale property—with this assumption at hand, they can truncate the last arm played to fit the budget without incurring any loss in the expected reward. However, such an idea fails without the martingale property, and these LPs have large integrality gaps (Appendix B).

A major drawback with the previous LP relaxations is that the constraints are *local* for each item/arm, i.e., they track the probability distribution over how long each item/arm is processed, and there is a global constraint on the total number of pulls/knapsack budget. Using such local constraints results in two different issues.

For the (correlated) stochastic knapsack problem, these LPs do not capture the case when all the items have high *contention*, i.e., they may all want to be played early in order to collect a huge profit from their large sizes. And for the general multi-armed bandit problem, we show that *no such localized solution* can be good since they do not capture the notion of *preempting* an arm, namely switching from one arm to another and possibly returning to the original arm later. Indeed, we show cases when any near-optimal strategy must repeatedly switch back-and-forth between arms (see Appendix C)—this is the crucial difference from previous work with the martingale property where there exist near-optimal strategies that never return to any arm [18, Lemma 2.1]. Hence our algorithm needs to make adaptive decisions, contrasting with previously existing index-based policies.

We resolve these issues in the following manner. Incorporating item cancellations into stochastic knapsack can be done by adapting the flow-like LPs from earlier works on MABs. To handle the issues of contention and preemption, we formulate a *global time-indexed* relaxation that forces the LP solution to commit each item to begin at a time, and places constraints on the maximum expected reward that can be obtained if the LP begins an item at a particular time. Furthermore, the time-indexing also enables our rounding scheme to extract information about when to preempt an arm and when to re-visit it based on the LP solution; in fact, these decisions will depend on the (random) outcomes of previous pulls, but the LP encodes the information for each eventuality. We hope that our fairly general techniques will be applicable to other problems in stochastic optimization.

Roadmap: In Section 2 we give a $O(1)$-approximation for the stochastic knapsack problem when rewards could

be correlated with the sizes; refer to the full version of the paper [21] for the general case with correlations and cancellations. Then in Section 3, we move on to the more general class of multi-armed bandit (MAB) problems. Again, for better clarity in presentation, we only present our MAB algorithm for the special case when the transition graph for each arm is an *arborescence*; the algorithm for arbitrary Markov chains is in the full version. However, we remark that this special case captures most of the core ideas of our rounding scheme. Likewise, while our MAB algorithms can also solve the stochastic knapsack problem, the latter motivates and gives insight into our techniques for MAB. Due to space constraints, we omit some proofs and refer the interested reader to the full version.

1.2. Related Work

Stochastic scheduling problems (in fact, even those with correlated rewards) have been long studied since the 1960s (e.g., [4], [25], [23]); however, there are fewer papers on approximation algorithms for such problems. [22], [14] consider stochastic knapsack problems with chance constraints: find the max-profit set that overflows the knapsack with probability at most p. However, their results hold for deterministic profits and specific size distributions. Minimizing average completion times with arbitrary job-size distributions was studied by [24], [27]. Most relevantly, Dean et al. [10], [9], [8] studied stochastic knapsack and packing; apart from algorithms for independent rewards/sizes, they show the problem with correlations to be PSPACE-hard. Bhalgat et al. [3] improve their approximation ratios, via giving a PTAS which is allowed to violate the capacity by a factor $(1+\epsilon)$. [6] study stochastic flow problems.

The general area of learning with costs is a rich and diverse one (see, e.g., [1], [12]). Approximation algorithms start with the work of Guha and Munagala [16], who gave LP-rounding algorithms for some of these problems. Further papers by these authors [20], [18] and by Goel et al. [15] give improvements, relate LP-based techniques and index-based policies, and also give new index policies. In particular, [18] considers metric switching costs and introduces a powerful Lagrangian-relaxation approach to solve it. In another recent work, [19] introduces an interesting generalization where that the outcome of pulling an arm is only observed some k steps later. Although motivated by different considerations, their algorithm also performs steps similar to Phases II and III of our MAB algorithm, and it would be interesting to see if there are deeper connections. Farias and Madan [11] study the MAB problem where multiple arms can be pulled simultaneously focusing explicitly on non-preemptive strategies. All the above papers assume some form of the martingale condition, and, with the exception of [19], produce non-preemptive strategies which are good approximations for the preemptive optimal solution. In contrast, solutions to the problem we consider here *have to* be preemptive to provide good approximations.

2. Correlated Stochastic Knapsack without Cancellation

We begin by considering the stochastic knapsack problem (StocK), when the item rewards may be correlated with its size. This generalizes the problem studied by Dean et al. [9] who assume that the rewards are independent of the size of the item. We first explain why the LP of [9] has a large integrality gap for our problem; this will naturally motivate our time-indexed formulation. We then present a simple randomized rounding algorithm which produces a non-adaptive strategy and show that it is an $O(1)$-approximation.

2.1. Problem Definitions and Notation

We are given a knapsack of total budget B and a collection of n stochastic items. For any item $i \in [1,n]$, we are given a probability distribution over $(\mathsf{size}, \mathsf{reward})$ pairs specified as follows: for each integer value of $t \in [1,B]$, the tuple $(\pi_{i,t}, R_{i,t})$ denotes the probability $\pi_{i,t}$ that item i has a size t, and the corresponding reward is $R_{i,t}$; The interpretation for $R_{i,t}$ is the conditional expected reward of item i given that its size is t. Note that the $(\mathsf{size}, \mathsf{reward})$ pairs for two different items are still independent of each other.

An adaptive algorithm can take the following actions at the end of each timestep; (i) an item may complete at a certain size (giving us the corresponding reward), and the algorithm may choose a new item to start, or (ii) the knapsack becomes full, at which point the algorithm stops, and the item being processed does not fetch any reward. The objective is to maximize the total expected reward obtained from all completed items.

2.2. LP Relaxation

The LP relaxation in [9] was (essentially) a knapsack LP where the sizes of items are replaced by the expected sizes, and the rewards are replaced by the expected rewards. While this was sufficient when an item's reward is fixed (or chosen randomly but independent of its size), we give an example in Appendix B where such an LP (and in fact, the class of more general LPs used for approximating MAB problems) would have a large integrality gap. As mentioned in Section 1.1, the reason why local LPs don't work is that there could be high contention for being scheduled early (i.e., there could be a large number of items which all fetch reward if they instantiate to a large size, but these events occur with low probability). In order to capture this contention, we write a global time-indexed LP relaxation, drawing inspiration from the LP in [8] for the stochastic knapsack problem with individual start deadlines.

The variable $x_{i,t} \in [0,1]$ indicates that item i is scheduled at (global) time t; S_i denotes the random variable for the size of item i, and $\mathsf{ER}_{i,t} = \sum_{s \le B-t} \pi_{i,s} R'_{i,s}$ captures the expected reward that can be obtained from item i *if it begins*

at time t; (no reward is obtained for sizes that cannot fit the remaining budget.)

$$\max \textstyle\sum_{i,t} \mathsf{ER}_{i,t} \cdot x_{i,t} \qquad (\mathsf{LP}_{\mathsf{NoCancel}})$$
$$\textstyle\sum_t x_{i,t} \le 1 \qquad \forall i \quad (2.1)$$
$$\textstyle\sum_{i,t' \le t} x_{i,t'} \cdot \mathbb{E}[\min(S_i, t)] \le 2t \qquad \forall t \in [B] \quad (2.2)$$
$$x_{i,t} \in [0,1] \qquad \forall t \in [B], \forall i \quad (2.3)$$

While the size of the above LP (and the running time of the rounding algorithm below) polynomially depend on B, i.e., pseudo-polynomial, it is possible to write a compact (approximate) LP and then round it; details on the polynomial time implementation appear in the full version [21].

Notice the constraints involving the *truncated random variables* in equation (2.2): these are crucial for showing the correctness of the rounding algorithm StocK-NoCancel. Furthermore, the ideas used here will appear subsequently in the MAB algorithm later; for MAB, even though we can't explicitly enforce such a constraint in the LP, we will be able to infer similar inequalities from a near-optimal LP solution.

Lemma 2.1 *The relaxation* $\mathsf{LP}_{\mathsf{NoCancel}}$ *is valid for the* StocK *problem when cancellations are not permitted, and has objective value* $\mathsf{LPOpt} \ge \mathsf{Opt}$*, where* Opt *is the expected profit of an optimal adaptive policy.*

Proof: Consider an optimal policy Opt and let $x^*_{i,t}$ denote the probability that item i is scheduled at time t. We show that $\{x^*\}$ is a feasible solution for the LP relaxation $\mathsf{LP}_{\mathsf{NoCancel}}$. It is easy to see that constraints (2.1) and (2.3) are satisfied; that the LP objective is at least the expected reward follows from a simple linearity of expectation. It remains to show that equations (2.2) are also satisfied. Consider some $t \in [B]$ and some run (over random choices of item sizes) of the optimal policy. Let $\mathbf{1}^{\mathsf{sched}}_{i,t'}$ be indicator variable that item i is scheduled at time t' and let $\mathbf{1}^{\mathsf{size}}_{i,s}$ be the indicator variable for whether the size of item i is s. Also, let L_t be the random variable indicating the last item scheduled at or before time t. Notice that L_t is the only item scheduled before or at time t whose execution may go over time t. Therefore, we get that

$$\sum_{i \ne L_t} \sum_{t' \le t} \sum_{s \le B} \mathbf{1}^{\mathsf{sched}}_{i,t'} \cdot \mathbf{1}^{\mathsf{size}}_{i,s} \cdot s \le t.$$

Including L_t in the summation and truncating the sizes by t, we immediately obtain

$$\sum_i \sum_{t' \le t} \sum_s \mathbf{1}^{\mathsf{sched}}_{i,t'} \cdot \mathbf{1}^{\mathsf{size}}_{i,s} \cdot \min(s,t) \le 2t.$$

Now, taking expectation (over all of Opt's sample paths) on both sides and using linearity of expectation we have

$$\sum_i \sum_{t' \le t} \sum_s \mathbb{E}\left[\mathbf{1}^{\mathsf{sched}}_{i,t'} \cdot \mathbf{1}^{\mathsf{size}}_{i,s}\right] \cdot \min(s,t) \le 2t.$$

However, because Opt decides whether to schedule an item before observing the size it instantiates to, we have that $\mathbf{1}^{\mathsf{sched}}_{i,t'}$ and $\mathbf{1}^{\mathsf{size}}_{i,s}$ are independent random variables; hence, the LHS above can be re-written as

$$\begin{aligned}
&\sum_i \sum_{t' \le t} \sum_s \Pr[\mathbf{1}^{\mathsf{sched}}_{i,t'} = 1 \wedge \mathbf{1}^{\mathsf{size}}_{i,s} = 1] \min(s,t) \\
&= \sum_i \sum_{t' \le t} \Pr[\mathbf{1}^{\mathsf{sched}}_{i,t'} = 1] \sum_s \Pr[\mathbf{1}^{\mathsf{size}}_{i,s} = 1] \min(s,t) \\
&= \sum_i \sum_{t' \le t} x^*_{i,t'} \cdot \mathbb{E}[\min(S_i, t)]
\end{aligned}$$

This completes the proof. ■

Now, given an optimal fractional solution, our rounding algorithm StocK-NoCancel (Algorithm 2.1) is very simple: (i) pick a random start deadline for each item according to the corresponding distribution in the optimal LP solution, and (ii) play the items in order of the (random) deadlines. To ensure that the budget is not violated, we also drop each item independently with some constant probability.

Algorithm 2.1 Algorithm StocK-NoCancel

1: for each item i, **assign** a random start deadline $D_i = t$ with probability $\frac{x^*_{i,t}}{4}$; with probability $1 - \sum_t \frac{x^*_{i,t}}{4}$, completely ignore item i ($D_i = \infty$ in this case).
2: **for** j from 1 to n **do**
3: Consider the item i which has the jth smallest deadline (and $D_i \ne \infty$)
4: **if** the items added so far to the knapsack occupy at most D_i space **then**
5: add i to the knapsack.

Notice that the rounding strategy obtains reward from all items which are not dropped and which do not fail (i.e. they can start being scheduled before the sampled deadline D_i in Step 1); we now bound the failure probability.

Lemma 2.2 *For every* i, $\Pr(i \text{ fails} \mid D_i = t) \le 1/2$.

Proof: Consider an item i and time $t \ne \infty$ and condition on the event that $D_i = t$. If $t = 0$, then by our choice of independent sampling in step 1, it is easy to see that, conditioned on $D_i = 0$, no other item has its start deadline to be 0 with probability at least $1/2$, and item i can begin by its deadline in this case (i.e., does not fail). So let us assume that $t > 0$. Consider the execution of the algorithm when it tries to add item i to the knapsack in steps 3-5. Now, let Z be a random variable denoting *how much of the interval* $[0,t]$ of the knapsack is occupied by previously scheduling items, at the time when i is considered for addition; since i does not fail when $Z < t$, it suffices to prove that $\Pr(Z \ge t) \le 1/2$.

For some item $j \ne i$, let $\mathbf{1}_{D_j \le t}$ be the indicator variable that $D_j \le t$; notice that by the order in which algorithm

StocK-NoCancel adds items into the knapsack, it is also the indicator that j was considered before i. In addition, let $\mathbf{1}^{\mathsf{size}}_{j,s}$ be the indicator variable that $S_j = s$. Now, if Z_j denotes the total amount of the interval $[0,t]$ that that j occupies, we have

$$Z_j \leq \mathbf{1}_{D_j \leq t} \sum_s \mathbf{1}^{\mathsf{size}}_{j,s} \min(s,t).$$

Now, using the independence of $\mathbf{1}_{D_j \leq t}$ and $\mathbf{1}^{\mathsf{size}}_{j,s}$, we have

$$\mathbb{E}[Z_j] \leq \mathbb{E}[\mathbf{1}_{D_j \leq t}] \cdot \mathbb{E}[\min(S_j,t)] = \tfrac{1}{4} \textstyle\sum_{t' \leq t} x^*_{j,t'} \cdot \mathbb{E}[\min(S_j,t)] \tag{2.4}$$

Since $Z = \sum_j Z_j$, we can use linearity of expectation and the fact that $\{x^*\}$ satisfies LP constraint (2.2) to get

$$\mathbb{E}[Z] \leq \tfrac{1}{4} \textstyle\sum_j \sum_{t' \leq t} x^*_{j,t'} \cdot \mathbb{E}[\min(S_j,t)] \leq \tfrac{t}{2} \ .$$

To conclude the proof of the lemma, we apply Markov's inequality to obtain $\Pr(Z \geq t) \leq 1/2$. ■

To complete the analysis, we use the fact that any item chooses a random start time $D_i = t$ with probability $x^*_{i,t}/4$; conditioned on this, it is added to the knapsack with probability at least $1/2$ from Lemma 2.2; in this case, we get expected reward at least $\mathsf{ER}_{i,t}$. The theorem below follows by a straightforward application of linearity of expectation.

Theorem 2.3 *The expected reward of algorithm StocK-NoCancel is at least* $\frac{1}{8}$ *of* LPOpt.

2.3. Handling Job Cancellations

In this section, we outline how to extend the above algorithm to handle premature job cancellations also. The high level idea is the following: we can create two copies of every item, the "early" version of the item, where we discard profits from any instantiation where the size of the item is more than $B/2$, and the "late" version of the item where we discard profits from instantiations of size at most $B/2$. For the first kind, we make use of the fact that contention for the early timesteps is not really an issue (since rewards are only for small size instantiations); that is, in this case, solutions which violate the budget by a factor of 2 can easily be converted into feasible solutions by sub-sampling items. Therefore we write flow-like LPs akin to those for MAB problems [20], and round them in a natural way to get a constant-factor approximation. For the second kind, we argue that *cancellations don't help*, and hence we can reduce it to StocK without cancellations. Indeed, notice that as an algorithm processes an item for its t^{th} timestep for $t < B/2$, it gets no more information about the reward than when starting (since all rewards are at large sizes). Furthermore, there is no benefit of canceling an item once it has run for at least $B/2$ timesteps – we can't get any reward by starting some other item.

3. MULTI-ARMED BANDITS

We now turn our attention to the more general Multi-Armed Bandits problem (MAB). In this framework, there are n *arms*: arm i has a collection of states denoted by $\mathcal{S}_i$, a starting state $\rho_i \in \mathcal{S}_i$; Without loss of generality, we assume that $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$ for $i \neq j$. Each arm also has a *transition graph* T_i, which is given as a polynomial-size (weighted) directed acyclic graph rooted at ρ_i. If there is an edge $u \to v$ in T_i, then the edge weight $p_{u,v}$ denotes the probability of making a transition from u to v if we play arm i when its current state is node u; hence $\sum_{v:(u,v)\in T_i} p_{u,v} = 1$. Each time we play an arm, we get a reward whose value depends on the state from which the arm is played. Let us denote the reward at a state u by r_u. Like mentioned in the introduction, we will assume that the graph T_i is in fact an *out-arborescence*, i.e., a tree. This is only done for clarity in exposition of the ideas involved, and the crux of our arguments also carry forward to the general case of DAGs. The details appear in the full version [21].

Problem Definition: For a concrete example, we consider the following budgeted learning problem. Each of the arms starts at the start state $\rho_i \in \mathcal{S}_i$. We get a reward from every state we reach, and the goal is to maximize the total expected reward, while making at most B plays across all arms. Our general framework can handle other problems (like the explore/exploit kind) as well; see the full version of the paper for a discussion.Even the Stochastic Knapsack problem considered in the previous section is a special case where each item corresponds to an arm; the evolution of the arm corresponds to the explored size for the item (and does not satisfy the martingale property). A formal reduction is given in the full version [21].

Notation: The transition graph T_i for arm i is an out-arborescence defined on the states $\mathcal{S}_i$ rooted at ρ_i. Let $\mathsf{depth}(u)$ of a node $u \in \mathcal{S}_i$ be the depth of node u in tree T_i, where the root ρ_i has depth 0. The unique parent of node u in T_i is denoted by $\mathsf{par}(u)$. Let $\mathcal{S} = \cup_i \mathcal{S}_i$ denote the set of all states in the instance, and $\mathsf{arm}(u)$ denote the arm to which state u belongs, i.e., the index i such that $u \in \mathcal{S}_i$. Finally, for $u \in \mathcal{S}_i$, we refer to the act of playing arm i when it is in state u as "playing state $u \in \mathcal{S}_i$", or "playing state u" if the arm is clear in context.

3.1. Global Time-indexed LP

Variable $z_{u,t} \in [0,1]$ indicates that the algorithm plays state $u \in \mathcal{S}_i$ at time t. For $u \in \mathcal{S}_i$ and time t, $w_{u,t} \in [0,1]$ indicates that arm i *first enters* state u at time t (happens if and only if the algorithm *played* $\mathsf{par}(u)$ at time $t-1$ and the arm jumped to state u). The following lemma bounds the LP cost.

$$\max \textstyle\sum_{u,t} r_u \cdot z_{u,t} \qquad (\mathsf{LP}_{\mathsf{mab}})$$

$$w_{u,t} = z_{\mathsf{par}(u),t-1} \cdot p_{\mathsf{par}(u),u} \quad \forall t,\ u \in \mathcal{S} \setminus \cup_i \{\rho_i\} \tag{3.5}$$

$$\textstyle\sum_{t' \leq t} w_{u,t'} \geq \sum_{t' \leq t} z_{u,t'} \quad \forall t,\ u \in \mathcal{S} \tag{3.6}$$

$$\sum_{u \in \mathcal{S}} z_{u,t} \leq 1 \quad \forall t \tag{3.7}$$

$$w_{\rho_i,1} = 1 \quad \forall i \tag{3.8}$$

Lemma 3.1 *The optimal LP reward* LPOpt *is at least* Opt*, the expected reward of an optimal adaptive strategy.*

3.2. The Rounding Algorithm

In order to best understand the motivation behind our rounding algorithm, it would be useful to go over the example which illustrates the necessity of preemption (repeatedly switching back and forth between the different arms) in Appendix C. At a high level, the rounding algorithm proceeds as follows. In Phase I, given an optimal LP solution, we decompose the fractional solution for each arm into a convex[1] combination of integral "strategy forests" (which are depicted in Figure 3.1): each of these tells us at what times to play the arm, and in which states to abandon the arm. Now, if we sample a random strategy forest for each arm from this distribution, we may end up scheduling multiple arms to play at some of the timesteps, and hence we need to resolve these conflicts. A natural approach might be to (i) sample a strategy forest for each arm, (ii) play these arms in some order, and (iii) for any arm follow the decisions (about whether to abort or continue playing) as suggested by the sampled strategy forest. But this is inherently non-preemptive and therefore, by the example in Appendix C, it must fail.

Another approach would be to play the sampled forests at their prescribed times; if multiple forests want to play at the same time slot, we round-robin over them. But now if some arm needs B contiguous steps to get to a state with very high reward, even a single play of some other arm in the middle would end up fetching us no reward!

Guided by these bad examples, we try to use continuity information in the sampled strategy forests—once we start playing some contiguous component (where the strategy forest plays the arm in every consecutive time step), we make decisions to switch arms only at the end of the component (i.e. at the leaves of the different trees in Figure 3.1(b)). The naïve implementation does not work, so we first alter the solution to make all strategy forests "nice"—loosely, these are forests where all the connected components of any strategy forest are separated by large gaps (Phase II). The final strategy is presented in Phase III, and the analysis appears in Section 3.2.3.

3.2.1. Phase I: Convex Decomposition: In this step, we decompose the fractional solution into a convex combination of "forest-like strategies" $\{\mathbb{T}(i,j)\}_{i,j}$, corresponding to the j^{th} forest for arm i. We first formally define what these forests look like: The j^{th} *strategy forest* $\mathbb{T}(i,j)$ for arm i is an assignment of values $\mathsf{time}(i,j,u)$ and $\mathsf{prob}(i,j,u)$ to each state $u \in \mathcal{S}_i$ such that:

(i) For $u \in \mathcal{S}_i$ and $v = \mathsf{par}(u)$, it holds that $\mathsf{time}(i,j,u) \geq 1 + \mathsf{time}(i,j,v)$, and

(ii) For $u \in \mathcal{S}_i$ and $v = \mathsf{par}(u)$, if $\mathsf{time}(i,j,u) \neq \infty$ then $\mathsf{prob}(i,j,u) = p_{v,u}\,\mathsf{prob}(i,j,v)$; else if $\mathsf{time}(i,j,u) = \infty$ then $\mathsf{prob}(i,j,u) = 0$.

We call a triple (i,j,u) a *tree-node* of $\mathbb{T}(i,j)$.[2] For any state $u \in \mathcal{S}_i$, the values $\mathsf{time}(i,j,u)$ and $\mathsf{prob}(i,j,u)$ denote the time at which arm i is played from state u, and the probability with which the arm is played from state u, according to strategy forest $\mathbb{T}(i,j)$.

Observe that the probability values are particularly simple: if $\mathsf{time}(i,j,u) = \infty$ then this strategy does not play the arm at u, and hence the probability is zero, else $\mathsf{prob}(i,j,u)$ is equal to the probability of reaching u over the random transitions according to T_i if we play the root with probability $\mathsf{prob}(i,j,\rho_i)$. Hence, we can compute $\mathsf{prob}(i,j,u)$ just given $\mathsf{prob}(i,j,\rho_i)$ and whether or not $\mathsf{time}(i,j,u) = \infty$. Note that the time values are not necessarily consecutive, plotting these on the timeline and connecting a state to its parents only when they are in consecutive timesteps (as in Figure 3.1) gives us forests, hence the name.

The algorithm to construct such a decomposition proceeds in rounds for each arm i; in a particular round, it "peels" off such a strategy as described above, and ensures that the residual fractional solution continues to satisfy the LP constraints, guaranteeing that we can repeat this process, which is similar to (but slightly more involved than) performing flow-decompositions.

Lemma 3.2 *Given a solution to (*$\mathsf{LP_{mab}}$*), there exists a collection of at most* $nB|\mathcal{S}|$ *strategy forests* $\{\mathbb{T}(i,j)\}$ *such that* $z_{u,t} = \sum_{j:\mathsf{time}(i,j,u)=t} \mathsf{prob}(i,j,u)$.[3] *Hence,* $\sum_{(i,j,u):\mathsf{time}(i,j,u)=t} \mathsf{prob}(i,j,u) \leq 1$ *for all* t.

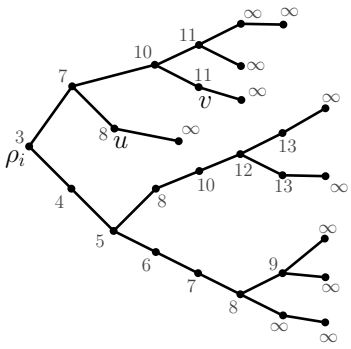
For any $\mathbb{T}(i,j)$, $\mathsf{prob}(\cdot)$ satisfies "preflow" conditions: the in-flow at any node v is at least the out-flow, namely $\mathsf{prob}(i,j,v) \geq \sum_{u:\mathsf{par}(u)=v} \mathsf{prob}(i,j,u)$, which leads to the following simple but crucial observation.

Observation 3.3 *For any arm* i*, for any set of states* $X \subseteq \mathcal{S}_i$ *such that no state in* X *is an ancestor of another in the transition tree* T_i*, and for any* $z \in \mathcal{S}_i$ *that is an ancestor of all states in* X*,* $\mathsf{prob}(i,j,z) \geq \sum_{x \in X} \mathsf{prob}(i,j,x)$*. More generally, if* Z *is a set of states such that for any* $x \in X$*, there exists* $z \in Z$ *such that* z *is an ancestor of* x*, we have* $\sum_{z \in Z} \mathsf{prob}(i,j,z) \geq \sum_{x \in X} \mathsf{prob}(i,j,x)$

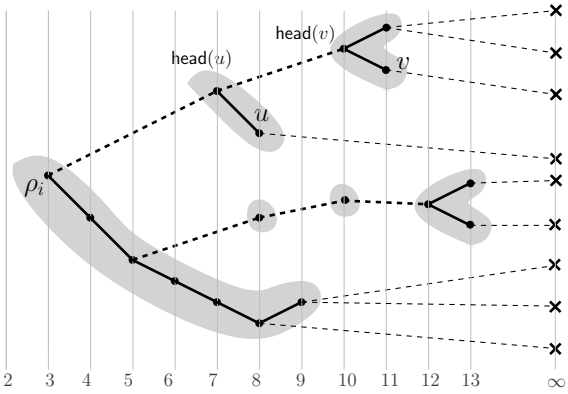
[1] Strictly speaking, we do not get convex combinations that sum to one; our combinations sum to $\sum_t z_{\rho_i,t}$, the value the LP assigned to pick to play the root of the arm over all possible start times, which is at most one.

[2] When i and j are understood from context, we identify the tree-node (i,j,u) with the state u.

[3] To reiterate, even though we call this a convex decomposition, the sum of the probability values of the root state of any arm is at most one by constraint 3.7, and hence the sum of the probabilities of the root over the decomposition could be less than one in general.



(a) Strategy forest: numbers are times



(b) Strategy forest shown on a timeline

Figure 3.1. Strategy forests and how to visualize them: grey blobs are connected components.

3.2.2. Phase II: Eliminating Small Gaps: While Appendix C shows that switching between arms is necessary, we also should not get "tricked" into switching arms during very short breaks taken by the LP strategy forest, e.g., if an arm of length $(B-1)$ with high reward at the end was played in two continuous segments with a small gap in the middle, we should not lose profit from this arm by starting some other arms' plays during the gap. We now handle this, by eliminating such small gaps between contiguous segments of the strategy forest.

The motivation for the procedure comes from the following proof argument: we would like to claim that our algorithm begins playing any component C before the start-time in the LP, with probability at least $1/2$. But the issue is that conditioned on playing C, we also get to know that all its ancestors have been played; and since other arms may have also been scheduled before C, the desired claim would be false. But if we ensure that the number of ancestors is small (say at most $t/2$, where t is the time when the LP begins playing C), this problem disappears—other arms use up to t plays on average (which we can make $t/2$ by sampling), leaving enough room for the ancestors' plays. This is precisely the condition we use to advance some components to fill small gaps.

Before we make this formal, here is some useful notation: Given $u \in \mathcal{S}_i$, let $\mathsf{Head}(i,j,u)$ be its ancestor node $v \in \mathcal{S}_i$ of least depth such that the plays from v through u occur in consecutive time values. More formally, the path $v = v_1, v_2, \ldots, v_l = u$ in T_i is such that $\mathsf{time}(i,j,v_{l'}) = \mathsf{time}(i,j,v_{l'-1}) + 1$ for all $l' \in [2,l]$. We also define the *connected component* of a node u, denoted by $\mathsf{comp}(i,j,u)$, as the set of all nodes u' such that $\mathsf{Head}(i,j,u) = \mathsf{Head}(i,j,u')$. Figure 3.1 shows the connected components and heads.

The *gap-filling* procedure works as follows: if a head state $v = \mathsf{Head}(i,j,u)$ is played at time $t = \mathsf{time}(i,j,v)$ s.t. $t < 2 \cdot \mathsf{depth}(v)$, then we "advance" the $\mathsf{comp}(i,j,v)$ and get rid of the gap between v and its parent (and recursively apply this rule)[4]. The interested reader may refer to the full version of the paper for a complete description of this procedure and the analysis.

By construction this guarantees that the components have large gaps between them. Additionally we show that the fractional number of plays made at any time t does not increase by too much due to these "advances". Intuitively this is because if for some time slot t we "advance" a set of components that were originally scheduled after t to now cross time slot t, these components moved because their ancestor paths (fractionally) used up at least $t/2$ of the time slots before t; since there are only a total of t time slots to be used up, there can be at most 2 units of components that were advanced across t. Hence, in the following, we assume that our $\mathbb{T}$'s satisfy the properties in the following lemma.

Lemma 3.4 *Algorithm GapFill produces a modified collection of $\mathbb{T}$'s such that*

(i) For each i,j,u such that $r_u > 0$, $\mathsf{time}(\mathsf{Head}(i,j,u)) \geq 2 \cdot \mathsf{depth}(\mathsf{Head}(i,j,u))$.

(ii) The total extent of plays at any time t, i.e., $\sum_{(i,j,u):\mathsf{time}(i,j,u)=t} \mathsf{prob}(i,j,u)$ is at most 3.

3.2.3. Phase III: Scheduling the Arms: After the above processing, the final algorithm is as follows: it samples a strategy forest from the collection $\{\mathbb{T}(i,j)\}_j$ for each arm i. Then, it picks an arm with the earliest connected component (i.e., the one with smallest $\mathsf{time}(\mathsf{Head}(i,j,u))$) that contains the current state (which is the root state to begin with), plays it to the end of the component, and repeats this step—note that we may switch out of an arm only if it jumps to a state played much later in time. Again we let the algorithm run as long as there is some active node, regardless of whether or not the budget is exceeded—however, we only count the profit from the first B plays in the analysis.

Observe that Steps 7-9 play a connected component of a strategy forest contiguously. In particular, this means that all $\mathsf{currstate}(i)$'s considered in Step 5 are head vertices of the

[4]The intuition is that such vertices have only a small gap in their play and should rather be played contiguously.

Algorithm 3.1 Scheduling the Connected Components: Algorithm AlgMAB

1: for arm i, **sample** strategy $\mathbb{T}(i,j)$ with probability $\frac{\mathsf{prob}(i,j,\rho_i)}{24}$; ignore arm i w.p. $1-\sum_j \frac{\mathsf{prob}(i,j,\rho_i)}{24}$.
2: let $A \leftarrow$ set of "active" arms which chose a strategy in the random process.
3: for each $i \in A$, **let** $\sigma(i) \leftarrow$ index j of the chosen $\mathbb{T}(i,j)$ and **let** $\mathsf{currstate}(i) \leftarrow$ root ρ_i.
4: **while** active arms $A \neq \emptyset$ **do**
5: **let** $i^* \leftarrow$ arm with state played earliest in the LP (i.e., $i^* \leftarrow \operatorname{argmin}_{i\in A}\{\mathsf{time}(i,\sigma(i),\mathsf{currstate}(i))\}$.
6: **let** $\tau \leftarrow \mathsf{time}(i^*,\sigma(i^*),\mathsf{currstate}(i^*))$.
7: **while** $\mathsf{time}(i^*,\sigma(i^*),\mathsf{currstate}(i^*)) \neq \infty$ **and** $\mathsf{time}(i^*,\sigma(i^*),\mathsf{currstate}(i^*)) = \tau$ **do**
8: **play** arm i^* at state $\mathsf{currstate}(i^*)$
9: **update** $\mathsf{currstate}(i^*)$ be the new state of arm i^*; **let** $\tau \leftarrow \tau+1$.
10: **if** $\mathsf{time}(i^*,\sigma(i^*),\mathsf{currstate}(i^*)) = \infty$ **then**
11: **let** $A \leftarrow A \setminus \{i^*\}$

corresponding strategy forests. These facts will be crucial in the analysis.

Lemma 3.5 *For arm i and strategy $\mathbb{T}(i,j)$, conditioned on $\sigma(i)=j$ after Step 1 of AlgMAB, the probability of playing state $u \in \mathcal{S}_i$ is $\mathsf{prob}(i,j,u)/\mathsf{prob}(i,j,\rho_i)$, where the probability is over the random transitions of arm i.*

The rest of the section proves that in expectation, we collect a constant factor of the LP reward of each strategy $\mathbb{T}(i,j)$ before running out of budget; the analysis is inspired by our StocK rounding procedure. We mainly focus on the following lemma.

Lemma 3.6 *Consider any arm i and strategy $\mathbb{T}(i,j)$. Then, conditioned on $\sigma(i)=j$ and on the algorithm playing state $u \in \mathcal{S}_i$, the probability that this play happens before time $\mathsf{time}(i,j,u)$ is at least $1/2$.*

Proof: Fix an arm i and an index j for the rest of the proof. Given a state $u \in \mathcal{S}_i$, let $\mathcal{E}_{iju}$ denote the event $(\sigma(i)=j) \wedge$ (state u is played). Also, let $\mathbf{v} = \mathsf{Head}(i,j,u)$ be the head of the connected component containing u in $\mathbb{T}(i,j)$. Let r.v. τ_u (respectively $\tau_{\mathbf{v}}$) be the actual time at which state u (respectively state $\mathbf{v}$) is played—these random variables take value ∞ if the arm is not played in these states. Then showing that $\Pr[\tau_u \le \mathsf{time}(i,j,u) \mid \mathcal{E}_{iju}] \ge 1/2$ is equivalent to showing that $\Pr[\tau_{\mathbf{v}} \le \mathsf{time}(i,j,\mathbf{v}) \mid \mathcal{E}_{iju}] \ge 1/2$, because the time between playing u and $\mathbf{v}$ is exactly $\mathsf{time}(i,j,u) - \mathsf{time}(i,j,\mathbf{v})$ since the algorithm plays connected components continuously (and we have conditioned on $\mathcal{E}_{iju}$). Hence, we can just focus on proving the latter inequality for vertex $\mathbf{v}$.

For brevity of notation, let $t_{\mathbf{v}} = \mathsf{time}(i,j,\mathbf{v})$. In addition, we define the order $\preceq$ to indicate which states can be played before $\mathbf{v}$. That is, again making use of the fact that the algorithm plays connected components contiguously, we say that $(i',j',v') \preceq (i,j,\mathbf{v})$ iff $\mathsf{time}(\mathsf{Head}(i',j',v')) \le \mathsf{time}(\mathsf{Head}(i,j,\mathbf{v}))$. Notice that this order is independent of the run of the algorithm; also it could be that $\mathsf{time}(i',j',v') > \mathsf{time}(i,j,\mathbf{v})$ yet $(i',j',v') \preceq (i,j,\mathbf{v})$.

For each arm $i' \neq i$ and index j', we define random variables $Z_{i'j'}$ used to count the number of plays that can possibly occur before the algorithm plays state $\mathbf{v}$. If $\mathbf{1}_{(i',j',v')}$ is the indicator variable of event $\mathcal{E}_{i'j'v'}$, define

$$Z_{i',j'} = \min\left(t_{\mathbf{v}}\ ,\ \textstyle\sum_{v':(i',j',v')\preceq(i,j,\mathbf{v})} \mathbf{1}_{(i',j',v')}\right)\ . \quad (3.9)$$

We truncate $Z_{i',j'}$ at $t_{\mathbf{v}}$ because we just want to capture how much time *up to* $t_{\mathbf{v}}$ is being used. Now consider the sum $Z = \sum_{i'\neq i}\sum_{j'} Z_{i',j'}$. Note that for arm i', at most one of the $Z_{i',j'}$ values will be non-zero in any scenario, namely the index $\sigma(i')$ sampled in Step 1. The first claim below shows that it suffices to consider the upper tail of Z, and show that $\Pr[Z \ge t_{\mathbf{v}}/2] \le 1/2$, and the second gives a bound on the conditional expectation of $Z_{i',j'}$.

Claim 3.7 $\Pr[\tau_{\mathbf{v}} \le t_{\mathbf{v}} \mid \mathcal{E}_{iju}] \ge \Pr[Z \le t_{\mathbf{v}}/2]$.

Claim 3.8

$$\mathbb{E}[Z_{i',j'} \mid \sigma(i')=j'] \le \sum_{v' \ s.t\ \mathsf{time}(i',j',v')\le t_{\mathbf{v}}} \frac{\mathsf{prob}(i',j',v')}{\mathsf{prob}(i',j',\rho_{i'})} + t_{\mathbf{v}} \left(\sum_{v' \ s.t\ \mathsf{time}(i',j',v')= t_{\mathbf{v}}} \frac{\mathsf{prob}(i',j',v')}{\mathsf{prob}(i',j',\rho_{i'})} \right)$$

Equipped with the above claims, we are ready to complete the proof of Lemma 3.6. Employing Claim 3.8 we get

$$\begin{aligned}
\mathbb{E}[Z] &= \sum_{i'\neq i}\sum_{j'} \mathbb{E}[Z_{i',j'}] \\
&= \sum_{i'\neq i}\sum_{j'} \mathbb{E}[Z_{i',j'} \mid \sigma(i')=j'] \cdot \Pr[\sigma(i')=j'] \\
&= \frac{1}{24}\sum_{i'\neq i}\sum_{j'} \Bigg\{ \sum_{v':\mathsf{time}(i',j',v')\le t_{\mathbf{v}}} \mathsf{prob}(i',j',v') + t_{\mathbf{v}} \Bigg(\sum_{v':\mathsf{time}(i',j',v')= t_{\mathbf{v}}} \mathsf{prob}(i',j',v') \Bigg) \Bigg\} \quad (3.10) \\
&= \frac{1}{24}\left(3\cdot t_{\mathbf{v}} + 3\cdot t_{\mathbf{v}}\right) \le \frac{1}{4} t_{\mathbf{v}}\ . \quad (3.11)
\end{aligned}$$

Equation (3.10) follows from the fact that each tree $\mathbb{T}(i,j)$ is sampled with probability $\frac{\mathsf{prob}(i,j,\rho_i)}{24}$ and (3.11) follows from Lemma 3.4. Applying Markov's inequality, we have that $\Pr[Z \ge t_{\mathbf{v}}/2] \le 1/2$. Finally, Claim 3.7 says that $\Pr[\tau_{\mathbf{v}} \le t_{\mathbf{v}} \mid \mathcal{E}_{iju}] \ge \Pr[Z \le t_{\mathbf{v}}/2] \ge 1/2$, which completes the proof. ■

Theorem 3.9 *The reward obtained by the algorithm AlgMAB is at least* $\Omega(\mathsf{LPOpt})$.

Proof: The theorem follows by a simple linearity of expectation. Indeed, the expected reward obtained from any state $u \in \mathcal{S}_i$ is at least $\sum_j \Pr[\sigma(i) = j] \Pr[\text{state } u \text{ is played} \mid \sigma(i) = j] \Pr[\tau_u \leq t_u | \mathcal{E}_{iju}] \cdot R_u \geq \sum_j \frac{\mathsf{prob}(i,j,u)}{24} \frac{1}{2} \cdot R_u$. Here, we have used Lemmas 3.5 and 3.6 for the second and third probabilities. But now we can use Lemma 3.2 to infer that $\sum_j \mathsf{prob}(i,j,u) = \sum_t z_{u,t}$; Making this substitution and summing over all states $u \in \mathcal{S}_i$ and arms i completes the proof. ■

Acknowledgments: We thank Kamesh Munagala and Sudipto Guha for useful conversations.

REFERENCES

[1] D. P. Bertsekas, *Dynamic programming and optimal control.*, 3rd ed. Athena Scientific, Belmont, MA, 2005.

[2] D. Bertsimas and A. J. Mersereau, "A learning approach for interactive marketing to a customer segment," *Operations Research*, vol. 55, no. 6, pp. 1120–1135, 2007.

[3] A. Bhalgat, A. Goel, and S. Khanna, "Improved approximation results for stochastic knapsack problems," in *SODA*, 2011.

[4] J. R. Birge and F. Louveaux, *Introduction to stochastic programming*. Springer-Verlag, 1997.

[5] D. Chakrabarti, R. Kumar, F. Radlinski, and E. Upfal, "Mortal multi-armed bandits," in *NIPS*, 2008, pp. 273–280.

[6] S. Chawla and T. Roughgarden, "Single-source stochastic routing," in *Proceedings of APPROX*, 2006, pp. 82–94.

[7] J. Coffman, E. G., L. Flatto, M. R. Garey, and R. R. Weber, "Minimizing expected makespans on uniform processor systems," *Adv. Appl. Prob.*, vol. 19, no. 1, pp. pp. 177–201, 1987.

[8] B. C. Dean, "Approximation algorithms for stochastic scheduling problems," Ph.D. dissertation, MIT, 2005.

[9] B. C. Dean, M. X. Goemans, and J. Vondrák, "Adaptivity and approximation for stochastic packing problems," in *SODA*, 2005, pp. 395–404.

[10] ——, "Approximating the stochastic knapsack problem: The benefit of adaptivity," *Math. Oper. Res.*, vol. 33, no. 4, pp. 945–964, 2008.

[11] V. F. Farias and R. Madan, "The irrevocable multiarmed bandit problem," *Oper. Res.*, vol. 59, no. 2, pp. 383–399, 2011.

[12] J. C. Gittins, *Multi-armed bandit allocation indices*. John Wiley & Sons Ltd., 1989.

[13] J. Gittins, K. Glazebrook, and R. Weber, *Multi-armed Bandit Allocation Indices*. Wiley Interscience, 2011.

[14] A. Goel and P. Indyk, "Stochastic load balancing and related problems," in *FOCS*, 1999, pp. 579–586.

[15] A. Goel, S. Khanna, and B. Null, "The ratio index for budgeted learning, with applications," in *SODA*, 2009, pp. 18–27.

[16] S. Guha and K. Munagala, "Approximation algorithms for budgeted learning problems," in *STOC*, 2007, pp. 104–113, full version as *Sequential Design of Experiments via Linear Programming*, http://arxiv.org/abs/0805.2630v1.

[17] ——, "Model-driven optimization using adaptive probes," in *SODA*, 2007, pp. 308–317, full version as *Adaptive Uncertainty Resolution in Bayesian Combinatorial Optimization Problems*, http://arxiv.org/abs/0812.1012v1.

[18] ——, "Multi-armed bandits with metric switching costs," in *ICALP*, 2009, pp. 496–507.

[19] S. Guha, K. Munagala, and M. Pal, "Iterated allocations with delayed feedback," *ArXiv*, vol. arxiv:abs/1011.1161, 2011.

[20] S. Guha, K. Munagala, and P. Shi, "On index policies for restless bandit problems," *CoRR*, vol. abs/0711.3861, 2007, http://arxiv.org/abs/0711.3861. Full version of *Approximation algorithms for partial-information based stochastic control with Markovian rewards* (FOCS'07), and *Approximation algorithms for restless bandit problems*, (SODA'09).

[21] A. Gupta, R. Krishnaswamy, M. Molinaro, and R. Ravi, "Approximation algorithms for correlated knapsacks and non-martingale bandits," *CoRR*, vol. abs/1102.3749, 2011.

[22] J. Kleinberg, Y. Rabani, and É. Tardos, "Allocating bandwidth for bursty connections," *SIAM J. Comput.*, vol. 30, no. 1, pp. 191–217, 2000.

[23] A. J. Kleywegt and J. D. Papastavrou, "The dynamic and stochastic knapsack problem with random sized items," *Oper. Res.*, vol. 49, pp. 26–41, January 2001.

[24] R. H. Möhring, A. S. Schulz, and M. Uetz, "Approximation in stochastic scheduling: the power of lp-based priority policies," *JACM*, vol. 46, no. 6, pp. 924–942, 1999.

[25] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. Prentice Hall, 1995.

[26] H. Robbins, "Some aspects of the sequential design of experiments," *Bull. AMS*, vol. 58, no. 5, pp. 527–535, 1952.

[27] M. Skutella and M. Uetz, "Scheduling precedence-constrained jobs with stochastic processing times on parallel machines," in *SODA*, 2001, pp. 589–590.

APPENDIX

1. Badness Due to Cancellations

We first observe that the LP relaxation for the StocK problem used in [10] has a large integrality gap in the model where cancellations are allowed, *even when the rewards are fixed for any item*. This was also noted in [8]. Consider the following example: there are n items, every item instantiates to a size of 1 with probability 0.5 or a size of $n/2$ with probability 0.5, and its reward is always 1. Let the total size of the knapsack be $B = n$. For such an instance, a good solution would cancel any item that does not terminate at size 1; this way, it can collect a reward of at least $n/2$ in expectation, because an average of $n/2$ items will instantiate with a size 1 and these will all contribute to the reward. On the other hand, the LP from [10] has value $O(1)$, since the mean size of any item is at least $n/4$. In fact, any strategy that does not cancel items will also accrue only $O(1)$ reward.

2. Badness Due to Correlated Rewards

Consider the following example: there are n items, every item instantiates to a size of 1 with probability $1 - 1/n$ or a size of n with probability $1/n$, and its reward is 1 only if its size is n, and 0 otherwise, and the knapsack size is $B = n$.

Clearly, any integral solution can fetch an expected reward of $1/n$ — if the first item it schedules instantiates to a large size, then it gives us a reward. Otherwise, no subsequent item can be fit within our budget if it instantiates to a large size. The issue with the existing LPs (even those for the more general MAB problem) is that the *arm-pull* transition probability constraints are ensured locally for each arm, and there is one global budget constraint. In our case, if we play each arm to completion individually, the expected size (i.e., number of pulls) it occupies is $1 \cdot (1-1/n) + n \cdot (1/n) \leq 2$. Therefore, such LPs can essentially accommodate $n/2$ items, fetching a total reward of $\Omega(1)$. Intuitively, what these LPs don't capture is that all items are competing to be pulled in the first time slot, and if we begin an item in any later time slot it fetches zero reward, which is why we consider a time-indexed LP in Section 2.

3. Badness Due to the Non-Martingale Property in MAB: The Benefit of Preemption

We now show that preemption is necessary in the case of MAB where the rewards do not satisfy the martingale property. This brings forward another key difference between our rounding scheme and earlier algorithms for MAB— the necessity of preempting arms is not an artifact of our algorithm/analysis but, rather, is unavoidable.

Consider the following instance. There are n identical arms, each of them with the following (recursively defined) transition tree starting at $\rho(0)$. When the root $\rho(j)$ is pulled for $j < m$, the following two transitions can happen:

(i) with probability $1/(n \cdot n^{m-j})$, the arm transitions to the "right-side", where if it makes $B - n(\sum_{k=0}^{j} L^k)$ plays, it will deterministically reach a state with reward n^{m-j}. All intermediate states have 0 reward.

(ii) with probability $1 - 1/(n \cdot n^{m-j})$, the arm transitions to the "left-side", where if it makes $L^{j+1} - 1$ plays, it will deterministically reach the state $\rho(j+1)$. No state along this path fetches any reward.

Finally, node $\rho(m)$ makes the following transitions when played: (i) with probability $1/n$, to a leaf state that has a reward of 1 and the arm ends there; (ii) with probability $1 - 1/n$, to a leaf state with reward of 0. (For the following calculations, assume that $B \gg L > n$ and $m \gg 0$.)

Preempting Solutions: We first exhibit a preempting solution with expected reward $\Omega(m)$. The strategy plays $\rho(0)$ of all the arms until one of them transitions to the "right-side", in which case it continues to play this until it fetches a reward of n^m. Notice that any root which transitioned to the right-side can be played to completion, because the number of pulls we have used thus far is at most n (only those at the $\rho(0)$ nodes for each arm), and the size of the right-side is exactly $B - n$. Now, if all the arms transitioned to the left-side, then it plays the $\rho(1)$ of each arm until one of them transitioned to the right-side, in which case it continues playing this arm and gets a reward of n^{m-1}. Again, any root $\rho(1)$ which transitioned to the right-side *can be played* to completion, because the number of pulls we have used thus far is at most $n(1+L)$ (for each arm, we have pulled the root $\rho(0)$, transitioned the walk of length $L-1$ to $\rho(1)$ and then pulled $\rho(1)$), and the size of the right-side is exactly $B - n(1+L)$. This strategy is similarly defined, recursively.

We now calculate the expected reward: if any of the roots $\rho(0)$ made a transition to the right-side, we get a reward of n^m. This happens with probability roughly $1/n^m$, giving us an expected reward of 1 in this case. If all the roots made the transition to the left-side, then at least one of the $\rho(1)$ states will make a transition to their right-side with probability $\approx 1/n^{m-1}$ in which case will will get reward of n^{m-1}, and so on. Thus, summing over the first $m/2$ such rounds, our expected reward is at least

$$\frac{1}{n^m} n^m + \left(1 - \frac{1}{n^m}\right) \frac{1}{n^{m-1}} n^{m-1} + \left(1 - \frac{1}{n^m}\right)\left(1 - \frac{1}{n^{m-1}}\right) \frac{1}{n^{m-2}} n^{m-2} + \ldots$$

Each term is $\Omega(1)$, hence the total expected reward is $\Omega(m)$.

Non-Preempting Solutions: Consider any non-preempting solution. Once it has played the first node of an arm and it has transitioned to the left-side, it has to irrevocably decide if it abandons this arm or continues playing. But if it has continued to play (and made the transition of $L-1$ steps), then it cannot get any reward from the right-side of $\rho(0)$ of any of the other arms, because $L > n$ and the right-side requires $B - n$ pulls before reaching a reward-state. Likewise, if it has decided to move from $\rho(i)$ to $\rho(i+1)$ on any arm, it cannot get *any* reward from the right-sides of $\rho(0), \rho(1), \ldots, \rho(i)$ on *any* arm due to budget constraints. Indeed, for any $i \geq 1$, to have reached $\rho(i+1)$ on any particular arm, it must have utilized $(1 + L - 1) + (1 + L^2 - 1) + \ldots + (1 + L^{i+1} - 1)$ pulls in total, which exceeds $n(1 + L + L^2 + \ldots + L^i)$ since $L > n$. Finally, notice that if the strategy has decided to move from $\rho(i)$ to $\rho(i+1)$ on any arm, the maximum reward that it can obtain is n^{m-i-1}, namely, the reward from the right-side transition of $\rho(i+1)$.

Using these properties, we observe that an optimal non-preempting strategy proceeds in rounds as described next.

Strategy at round i: Choose a set N_i of n_i available arms and play them as follows: pick one of these arms, play until reaching state $\rho(i)$ and then play once more. If there is a right-side transition before reaching state $\rho(i)$, discard this arm since there is not enough budget to play until reaching a state with positive reward. If there is a right-side transition at state $\rho(i)$, play this arm until it gives reward of n^{m-i}. If there is no right-side transition and there is another arm in N_i which is still to be played, discard the current arm and pick the next arm in N_i.

In round i, at least $\max(0, n_i - 1)$ arms are discarded, hence $\sum_i n_i \leq 2n$. The expected reward is hence at most

$$\frac{n_1}{n \cdot n^m} n^m + \frac{n_2}{n \cdot n^{m-1}} n^{m-1} + \ldots + \frac{n_m}{n} \leq 2\,.$$