Improvements and Generalizations of Stochastic Knapsack and Multi-Armed Bandit Approximation Algorithms: Extended Abstract

Will Ma*

Abstract

The multi-armed bandit (MAB) problem features the classical tradeoff between exploration and exploitation. The input specifies several stochastic *arms* which evolve with each *pull*, and the goal is to maximize the expected reward after a fixed budget of pulls. The celebrated work of Gittins et al., surveyed in [8], presumes a condition on the arms called the *martingale assumption*. In [9], A. Gupta et al. obtained an LP-based $\frac{1}{48}$-approximation for the problem with the martingale assumption removed. We improve the algorithm to a $\frac{4}{27}$-approximation, with simpler analysis. Our algorithm also generalizes to the case of MAB superprocesses with (stochastic) multi-period actions. This generalization captures the framework introduced by Guha and Munagala in [11], and yields new results for their budgeted learning problems.

Also, we obtain a $(\frac{1}{2}-\varepsilon)$-approximation for the variant of MAB where preemption (playing an arm, switching to another arm, then coming back to the first arm) is not allowed. This contains the stochastic knapsack problem of Dean, Goemans, and Vondrák in [6] with correlated rewards, where we are given a knapsack of fixed size, and a set of jobs each with a joint distribution for its size and reward. The actual size and reward of a job can only be discovered in real-time as it is being scheduled, and the objective is to maximize expected reward before the knapsack size is exhausted.

Our $(\frac{1}{2}-\varepsilon)$-approximation improves the $\frac{1}{16}$ and $\frac{1}{8}$ approximations in [9] for correlated stochastic knapsack with cancellation and no cancellation, respectively, providing to our knowledge the first tight algorithm for these problems that matches the integrality gap of 2. We sample probabilities from an exponential-sized dynamic programming solution, whose existence is guaranteed by an LP projection argument. We hope this technique can also be applied to other dynamic programming problems which can be projected down onto a small LP.

1 Introduction

The multi-armed bandit (MAB) problem is the following: there are some Markov chains (*arms*), each of which only evolve to the next node and return some reward when you *pull* that arm; the controller has to allocate a fixed number of pulls amongst the arms to maximize expected reward. We use the word *node* instead of *state* to avoid confusion with the notion of a state in dynamic programming. The reward returned by the next pull of an arm depends on the current node of the arm. When an arm is pulled, the controller sees the outcome and can use that information to make the next decision. Let $p_{u,v}$ denote the probability of transitioning from node u to v, when the arm is pulled from node u. Furthermore, each node u has a reward $r_u \geq 0$ that is accrued whenever the arm is pulled from that node. Since we are only concerned with maximizing expected reward, we can assume r_u is a constant that does not depend on the transition that is taken from u.

The martingale assumption states that for all nodes u, $r_u = \sum_v p_{u,v} \cdot r_v$. The motivation comes from budgeted learning problems, where the node of an arm represents the updated prior distribution on some unknown quantity. In [9], A. Gupta et al. study the multi-amed bandit problem with the martingale assumption removed. The introduction of their paper discusses the removal of the martingale assumption, which we do not repeat here. The problem becomes intractable even for special cases (for details, see the introduction of [12] and the references therein), so they approach the problem from the perspective of approximation algorithms. They obtain an LP-based $\frac{1}{48}$-approximation for the multi-armed bandit problem. We improve the algorithm to a $\frac{4}{27}$-approximation, by eliminating the need for their *convex decomposition* and *gap filling* operations.

Furthermore, our algorithm allows for a more general family of inputs, in two ways. First, we allow transitions on the Markov chains that consume more than one pull worth of budget. We can think of these transitions as having a non-unit *processing time*. The processing times can be stochastic, and correlated with the node transition that takes place. The rewards can be accrued upon pulling the node, or only accrued if the processing time completes before the time budget runs out. The applications of such a generalization to US Air Force jet maintenance have recently been considered in [14], where it is referred to as *multi-period actions*. However, under this generalization, we can only guarantee an approximation factor of $\frac{1}{12}$.

*willma353@gmail.com, Operations Research Center, Massachusetts Institute of Technology. Supported in part by the NSERC PGS-D Award, NSF grant CCF-1115849, and ONR grants N00014-11-1-0053 and N00014-11-1-0056.

The second generalization is that we allow each arm to be a Markov decision process; such a problem is referred to as an *MAB superprocess* [8]. Now, when the controller pulls an arm, they have a choice of actions for the node that arm is on, each of which results in a different joint distribution on reward, processing time, and transition taken. After this generalization, the budgeted learning framework introduced in [11] becomes a special case of our problem, where exploiting just corresponds to a different action. They have a $\frac{1}{4}$-approximation for their problem under the martingale assumption; our $\frac{4}{27}$-approximation works in general.

After these generalizations, we call the problem *MAB superprocess with multi-period actions and preemption*. We say "with preemption" because we also consider the variant where preemption is disallowed, namely, after we start pulling an arm, if we ever stop pulling it, then we can never pull it again. When the ability to preempt is removed, we obtain a tight $(\frac{1}{2}-\varepsilon)$-approximation with runtime polynomial in the input and $\frac{1}{\varepsilon}$. This algorithm is stronger because its analysis doesn't resort to Markov bounds. [9] shows that for general Markov chains, preemption is indispensable — preemptive policies can be an unbounded factor better than the best non-preemptive policies. Nonetheless, this variant of the problem still contains the stochastic knapsack problem with correlated rewards — both with and without cancellation.

The stochastic knapsack problem was introduced by Dean et al. in 2004 (see [6] for the journal version). We are to schedule some jobs under a fixed time budget, each with a stochastic reward and processing time whose distribution is known beforehand. We sequentially choose which job to do next, only discovering its length and reward in real-time as it is being processed. The objective is to maximize expected reward before the time budget is spent.

Throughout [6], the authors assume uncorrelated rewards — that is, the reward of a job is independent of its length. A. Gupta et al. obtain a $\frac{1}{8}$-approximation for stochastic knapsack with this assumption removed. Furthermore, they obtain a $\frac{1}{16}$-approximation for the variant where jobs can be cancelled at any time. Both of these variants are special cases of *MAB superprocess with multi-period actions and no preemption*: the former because we can treat each job as an arm that is a single transition (with stochastic processing time and reward upon completion); the latter because we can treat each job as an arm that is a caterpillar tree (see the example at the beginning of Section 4 in [9]). Therefore, our $(\frac{1}{2}-\varepsilon)$-approximation works for both of these problems.

However, it is important to mention that our result, as well as the results of A. Gupta et al. in [9], require the job sizes and budget to be given in unary, since these results use the time-indexed LP. It appears that this LP is necessary whenever correlation is allowed — Gupta et al. show that the non-time-indexed LP can be off by an arbitrarily large factor. This dichotomy is further reinforced by the recent result of Li and Yuan in [15] — they obtain a $(\frac{1}{2}-\varepsilon)$-approximation for binary stochastic knapsack with cancellation, but require the job rewards to be independent of their sizes. Gupta et al. outline techniques for approximating the time-indexed LP if the job sizes and budget are given in binary, albeit losing some approximation factor. Nonetheless, in this paper, we will always think of processing times as discrete hops on a Markov Chain, given in unary. Note that Dean et al. obtain stronger hardness results than those of Guha and Munagala when the sizes are given in binary (for details, see the introduction of [6] and the references therein).

We describe some of the techniques in our $(\frac{1}{2}-\varepsilon)$-approximation. We write the exponential-sized LP corresponding to dynamic programming, along with a polynomial LP relaxation that is a projection of this LP onto a subspace. This motivates an exponential algorithm that can convert a feasible solution of the relaxation into a feasible solution of the large LP, with all the probabilities scaled in half. We want to convert this solution of the large LP into a policy, but can't compute some of the probabilities in polynomial time. However, we can sample these probabilities in real-time as we execute the algorithm, in a way that doesn't cause error propagation. For sampling basics, see Chapter 11 of the textbook [18] of Motwani and Raghavan. We leave it as an open problem to eliminate the need for sampling (and thus the ε error) from our algorithm.

We define the *projection gap* of an LP relaxation to be the supremum, over all instances, of the optimum of the relaxation divided by the optimum of the exponential-sized LP[1]. Our exponential algorithm upper bounds the projection gap at 2, and we come up with a matching lower bound. This shows that we cannot do better than a $\frac{1}{2}$-approximation using the same LP relaxation. Projection gap is the analogue of integrality gap when the solution of the NP-hard problem is given by an exponential-sized linear program, instead of an integer program. For the variant with preemption, we still cannot construct any example showing the projection gap is greater than 2 — in fact we conjecture that the projection gap is 2 and the upper bound of $\frac{27}{4}$

[1]For our maximization problems, we have used α-approximation to mean a polynomial-time algorithm that obtains at least α of the optimal reward, with $\alpha < 1$. However, for projection gap, we will follow the convention for integrality gaps, where all factors are greater than 1.

can be improved.

1.1 Related Work The stochastic knapsack problem has generated a wealth of interest since it was introduced in 2004. Part of the appeal of the problem is the strength of *non-adaptive* algorithms — algorithms that fix the sequence of jobs to be scheduled before seeing any realizations of job lengths. Some results in the area are the $(\frac{1}{2} - \varepsilon)$-approximation in [4], and the bicriteria $(1 - \varepsilon)$-approximation in [2] that uses $(1 + \varepsilon)$ as much time. Recently, [15] matched the latter result with Poisson approximation, and also achieved this for the case of correlated rewards with cancellation.

Stochastic knapsack also triggered a long line of work by Guha, Munagala, and various co-authors in the explore-exploit budgeted learning framework, starting with [11] and most recently summarized in [12]. They discuss many variations (switching costs, simultaneous plays, delayed feedback) that we do not consider here. Looking at older work, we point the reader interested in the area of multi-armed bandit learning problems and Gittins indices to [8]. Related problems involving optimal control, stopping time theory, discounted rewards, and infinite time horizons are described thoroughly in [1]. From a stochastic scheduling perspective, the stadard textbook is [19].

Looking at more recent work, a group to the authors similar to that of [9] provide a $(\log \log B)$-approximation for a stochastic orienteering problem in [10]. Even more recently, this adaptive stochastic framework has been introduced to online matching problems in [17]. Another problem with this flavor can be found in [3] — in fact we use one of their lemmas in our analysis. All of these papers deal strictly with expected reward. There has been recent progress on the variant of stochastic knapsack where the objective is to maximize the probability of achieving a target reward; see [13, 5].

1.2 Organization of Paper This is a condensed version of the paper.[2] In Section 2, we describe our algorithm for the vanilla MAB problem (with preemption) and prove it is a $\frac{4}{27}$-approximation. The generalization to *MAB superprocess with multi-period actions and preemption*, along with the new analysis that only yields a $\frac{1}{12}$-approximation, use similar ideas and are provided in the full version. In Section 3, we obtain a $(\frac{1}{2} - \varepsilon)$-approximation for *correlated stochastic knapsack with no cancellation* and provide the example lower bounding the projection gap at 2. The generalization to *MAB superprocess with multi-period actions and no preemption* uses similar ideas and is provided in the full version.

[2]The full version of this paper is available on arXiv.org as [16].

At the end of each section, we give an overview of the how the generalizations follow in the full version.

1.3 Comparison of Results

MAB	$\frac{1}{48}$	[9]
MAB	$\frac{4}{27}$	Sec. 2
Budgeted learning w/ martingale assump.	$\frac{1}{4}$	[11]
MAB superprocess	$\frac{4}{27}$	full ver. [16]
MAB superprocess w/ multi-period actions	$\frac{1}{12}$	full ver. [16]

Binary SK	$\frac{1}{2} - \varepsilon$	[4]
Binary SK w/ canc.	$\frac{1}{2} - \varepsilon$	[15]
Unary Corr. SK	$\frac{1}{8}$	[9]
Unary Corr. SK w/ canc.	$\frac{1}{16}$	[9]
Unary Corr. SK	$\frac{1}{2} - \varepsilon$	Sec. 3
Unary Corr. SK w/canc.	$\frac{1}{2} - \varepsilon$	full ver. [16]
MAB superprocess w/ multi-period actions no preemption	$\frac{1}{2} - \varepsilon$	full ver. [16]

1.4 Acknowledgements The author would like to thank advisor Michel Goemans for insightful discussions and suggestions on the presentation of this paper. Useful remarks from several anonymous referees have also improved the presentation of this paper.

2 $\frac{4}{27}$-approximation for MAB

2.1 Preliminaries The multi-armed bandit problem is the following: there are n Markov chains, or arms, indexed by i. Let ρ_i denote the root node for arm i. Let $p_{u,v}$ denote the transition probability from node u to v and let $r_u \geq 0$ denote the reward on node u. Note that for all nodes u, $\sum_v p_{u,v} = 1$.

We also assume we have converted each rooted Markov chain into a layered, acyclic digraph. That is, for each arm i there exists a function depth from its nodes to $\{0, 1, \ldots\}$ such that $\mathsf{depth}(\rho_i) = 0$ and all transitions $p_{u,v} > 0$ satisfy $\mathsf{depth}(v) = \mathsf{depth}(u) + 1$.

This can be done by expanding each node in the original graph into a time-indexed copy of itself for each time step — we refer to [9] for the standard reduction. Note that nodes can still have multiple parents. It is impossible to eliminate this (by converting the arms into directed trees) without exponential blow-up in the worst case.

For convenience, our notation will not assume a terminal time step. Of course, we will not actually have any Markov chains with infinitely many states at arbitrarily large depths — all of our algorithms will have a polynomial depth at which we can cut off the Markov chains, resulting in only polynomial blow-up from this reduction.

After all the reductions, let $\mathcal{S}_i$ denote the set of nodes of arm i, and let $\mathcal{S} = \bigcup_{i=1}^{n} \mathcal{S}_i$. Let $\mathsf{Par}(u) = \{v \in \mathcal{S} : p_{u,v} > 0\}$, the set of *parents* of node u. Each Markov chain i starts on its root node, ρ_i. At each time step, we choose an arm to pull, getting reward r_u, where u is the node the arm was on. We realize the transition that is taken before deciding which arm to pull next. The objective is to maximize the expected reward accrued after a budget of B time steps.

2.2 Linear Programming Formulations Algorithms for this problem are described in the form of an adaptive *policy*, a specification of which arm to pull for each state we could be in. A state in this case is determined by the following: the node each arm is on, and the time step we are at (for convenience, we will allow ourselves to not pull any arm at a time step, so the time elapsed cannot be deduced from the nodes the arms are on and must be included in the state information). The optimal policy could theoretically be obtained by dynamic programming (DP), but of course there are exponentially many states. However, we can still write the Bellman state-updating equations to get an LP whose feasible region corresponds to exactly the set of admissible policies. After adding in the objective function of maximizing expected reward, solving this exponential-sized LP would be equivalent to solving the problem via DP.

We would like to write a polynomial LP relaxation for the large DP. We keep track of the probabilities on the nodes of each arm individually without considering their joint distribution, so that we no longer have exponentially many variables. Let $s_{u,t}$ be the probability arm i is on node u at the beginning of time t. Let $x_{u,t}$ be the probability we play node u at time t. For a positive integer N, let $[N]$ denote $\{1, \ldots, N\}$. Now we define (LP):

$$\max \sum_{u \in \mathcal{S}} r_u \sum_{t=1}^{B} x_{u,t} \tag{2.1}$$

$$\sum_{u \in \mathcal{S}} x_{u,t} \leq 1, \ t \in [B] \tag{2.2}$$

$$x_{u,t} \leq s_{u,t}, \ u \in \mathcal{S}, \ t \in [B] \tag{2.3}$$

$$x_{u,t} \geq 0, \ u \in \mathcal{S}, \ t \in [B] \tag{2.4}$$

$$s_{\rho_i,1} = 1, \ i \in [n] \tag{2.5}$$

$$s_{u,1} = 0, \ u \in \mathcal{S} \setminus \{\rho_1, \ldots, \rho_n\} \tag{2.6}$$

$$s_{u,t} = s_{u,t-1} - x_{u,t-1} + \sum_{v \in \mathsf{Par}(u)} x_{v,t-1} \cdot p_{v,u}, \quad t > 1, \ u \in \mathcal{S} \tag{2.7}$$

The only decision variables are the x's; there are as many equations in (2.5)-(2.7) as there are s variables. Our relaxation looks different than that of [9] but can be shown to be equivalent.

The key fact is that the optimal value of (LP) is an upper bound on the performance of any adaptive policy. Formally, let $\mathsf{OPT}_{\mathtt{DP}}$, $\mathsf{OPT}_{\mathtt{LP}}$ denote the optimum values of the large DP, and (LP), respectively. Then we have that $\mathsf{OPT}_{\mathtt{DP}} \leq \mathsf{OPT}_{\mathtt{LP}}$. The formal proof and the DP itself are highly technical and written out in the full version, but the idea is that a feasible solution of the DP can be projected onto a feasible solution of (LP). Similar proofs have appeared in [6, 9]; ours is a generalization put in the context of an exponential-sized DP.

Knowing that $\mathsf{OPT}_{\mathtt{LP}}$ is at least the optimum, we can now use the optimal solution of (LP) to get approximation algorithms. Such an LP-based algorithm would provide an upper bound on the supremum of $\frac{\mathsf{OPT}_{\mathtt{LP}}}{\mathsf{OPT}_{\mathtt{DP}}}$ over all instances of the multi-armed bandit problem. We term this quantity the *projection gap* of (LP); it is the analogue of *integrality gap* when the solution of the NP-complete problem is given by an exponential-sized LP instead of an integer program, and is sometimes erroneously referred to as such.

THEOREM 2.1. *There is an* (LP)*-based* $\frac{4}{27}$*-approximation algorithm for the multi-armed bandit problem. Therefore, the Projection Gap of* (LP) *is at most* $\frac{27}{4}$.

Proof. Our algorithm will maintain a priority index for each arm, telling us when the algorithm will try to play that arm again. If an arm is on node u with priority index t, then we say the arm is in *status* (u, t), with

$t = \infty$ indicating we are never playing that arm again. We need to define how to update these indices. Fix an optimal solution to (LP), $\{x_{u,t}, s_{u,t}\}$. Our goal is to make it so that the unconditional probability an arm is ever in status (u,t) is $\frac{x_{u,t}}{3}$, for all $u \in \mathcal{S}$, $t < \infty$.

Therefore, we initialize each arm i to status (ρ_i, t) with probability $\frac{x_{\rho_i,t}}{3}$ for all $t \in [B]$, and status (ρ_i, ∞) with probability $1 - \sum_{t=1}^{B} \frac{x_{\rho_i,t}}{3}$. Now, suppose we play an arm from status (v,t') and transition to some node u. We need to decide which status (u,t) to put the arm into. We would like to choose a $t > t'$, while at the same time considering the aforementioned goal. Let $q_{v,t',u,t}$ be the probability we put the arm into status (u,t), conditioned on us playing the arm from status (v,t') and arriving at u. The following lemma says we can find q's that satisfy both conditions:

LEMMA 2.1. *Suppose we are given the x's of a feasible solution to* (LP). *Then we can find $\{q_{v,t',u,t} : v \in \mathsf{Par}(u), t' < t\}$ in polynomial time such that*

$$\sum_{t > t'} q_{v,t',u,t} \le 1 \tag{2.8}$$

for all $v \in \mathcal{S}, t' \in [B], x_{v,t'} > 0, p_{v,u} > 0$ and

$$\sum_{v \in \mathsf{Par}(u)} \big(\sum_{t' < t} x_{v,t'} \cdot q_{v,t',u,t} \big) \cdot p_{v,u} = x_{u,t} \tag{2.9}$$

for all $u \in \mathcal{S} \setminus \{\rho_1, \dots, \rho_n\}, t \in [B], x_{u,t} > 0$. For notational convenience, these equations contain some undefined q's, which are assumed to be 0.

This lemma is intuitively simple, so we leave its mechanical proof to the full version. It is the replacement for *convex decomposition* from [9]. Note that finding these q's is an independent problem for each arm; we don't need the fact that x satisfies (2.2), the one set of constraints of (LP) that combines the arms.

After finding these q's, we define $q_{v,t',u,\infty}$ to be the difference in inequality (2.8), for all $v \in \mathcal{S}$, $t' \in [B]$, $x_{v,t'} > 0$, $p_{v,u} > 0$. $q_{v,t',u,\infty}$ is the probability we completely abandon the arm after playing it from status (v,t') and arriving at u.

Now we are ready to describe the algorithm. We initialize each arm i to status (ρ_i, t) with the aforementioned probabilities, for $t = 1, \dots, B, \infty$. Then:

1. While there exists an arm with priority not ∞, play an arm with the lowest priority (breaking ties arbitrarily) until it arrives at a status (u,t) such that $t \ge 2 \cdot \mathsf{depth}(u)$ ($t = \infty$ would suffice).

2. Repeat until all arms have priority ∞.

Of course, we are constrained by a budget of B time steps, but it will simplify the analysis to assume our algorithm finishes all the arms and collects reward only for plays up to time B. Under this assumption, the statuses an arm goes through is independent of the outcomes on all other arms; the inter-dependence only affects the order in which the statuses of the arms are played (and thus affects which nodes obtain reward).

For all $i \in [n]$, $u \in \mathcal{S}_i$, $t \in [B]$, let $\mathsf{time}(u,t)$ be the random variable for the time step at which our algorithm plays arm i from status (u,t), with $\mathsf{time}(u,t) = \infty$ if our algorithm never plays arm i from status (u,t). Then $\Pr[\mathsf{time}(\rho_i,t) < \infty] = \frac{x_{\rho_i,t}}{3}$ for all $i \in [n]$, $t \in [B]$. If u is not a root node, we can induct on $\mathsf{depth}(u)$ to prove that for all $t \in [B]$:

$$\begin{aligned}
&\Pr[\mathsf{time}(u,t) < \infty] \\
&= \sum_{v \in \mathsf{Par}(u)} \sum_{t' < t} \Pr[\mathsf{time}(v,t') < \infty] \cdot p_{v,u} \cdot q_{v,t',u,t} \\
&= \sum_{v \in \mathsf{Par}(u)} \big(\sum_{t' < t} \frac{x_{v,t'}}{3} \cdot q_{v,t',u,t} \big) \cdot p_{v,u} \\
&= \frac{x_{u,t}}{3}
\end{aligned}$$

where the final equality follows from Lemma 2.1. For an event $\mathcal{A}$, let $\mathbb{1}_{\mathcal{A}}$ denote the indicator random variable for event $\mathcal{A}$. The expected reward obtained by our algorithm is

$$\begin{aligned}
&\mathbb{E}\big[\sum_{u \in \mathcal{S}} r_u \sum_{t=1}^{B} \mathbb{1}_{\{\mathsf{time}(u,t) \le B\}} \big] \\
&= \sum_{u \in \mathcal{S}} r_u \sum_{t=1}^{B} \big(\mathbb{E}[\mathbb{1}_{\{\mathsf{time}(u,t) \le B\}} \mid \mathsf{time}(u,t) < \infty] \cdot \\
&\quad \Pr[\mathsf{time}(u,t) < \infty] \big) \\
&= \sum_{u \in \mathcal{S}} r_u \sum_{t=1}^{B} \Pr[\mathsf{time}(u,t) \le B \mid \mathsf{time}(u,t) < \infty] \cdot \frac{x_{u,t}}{3}
\end{aligned}$$

where the first equality uses the fact that $\mathsf{time}(u,t) \le B \implies \mathsf{time}(u,t) < \infty$. We will prove $\Pr[\mathsf{time}(u,t) \le B \mid \mathsf{time}(u,t) < \infty] \ge \frac{4}{9}$ for an arbitrary $i \in [n]$, $u \in \mathcal{S}_i$, $t \in [B]$. It suffices to prove $\Pr[\mathsf{time}(u,t) \le t \mid \mathsf{time}(u,t) < \infty] \ge \frac{4}{9}$, since $t \le B$.

Case 1. Suppose $t \ge 2 \cdot \mathsf{depth}(u)$. We will prove that conditioned on the event $\{\mathsf{time}(u,t) < \infty\}$ occuring, event $\{\mathsf{time}(u,t) > t\}$ occurs with probability at most $\frac{5}{9}$.

Note that every node v can have at most one t' such that $\mathsf{time}(v,t') < \infty$; let $\mathsf{time}(v)$ be equal to $\mathsf{time}(v,t')$

for this t' (and equal ∞ if $\mathsf{time}(v,t') = \infty$ for all $t' \in [B]$). The nodes v that are played before u are those with $\mathsf{time}(v) < \mathsf{time}(u,t)$. Since our algorithm makes a play at every time step, $\mathsf{time}(u,t) > t$ if and only if there are t or more nodes v such that $\mathsf{time}(v) < \mathsf{time}(u,t)$. The $\mathsf{depth}(u)$ ancestors of u are guaranteed to satisfy this.

Hence the event $\{\mathsf{time}(u,t) > t\}$ implies $\{\mathsf{depth}(u) + \sum_{v \in \mathcal{S} \setminus \mathcal{S}_i} \mathbb{1}_{\{\mathsf{time}(v) < \mathsf{time}(u,t)\}} \geq t\}$. But $t \geq 2 \cdot \mathsf{depth}(u)$, so this implies $\{\sum_{v \in \mathcal{S} \setminus \mathcal{S}_i} \mathbb{1}_{\{\mathsf{time}(v) < \mathsf{time}(u,t)\}} \geq \frac{t}{2}\}$. Now, whether the sum is at least $\frac{t}{2}$ is unchanged if we exclude from the sum all v such that $\mathsf{depth}(v) \geq \frac{t}{2}$. Indeed, if any such v satisfies $\mathsf{time}(v) < \mathsf{time}(u,t)$, then all of its ancestors also would, making the sum at least $\frac{t}{2}$ by themselves. So the last event is equivalent to $\{\sum_{v \in \mathcal{S} \setminus \mathcal{S}_i : \mathsf{depth}(v) < \frac{t}{2}} \mathbb{1}_{\{\mathsf{time}(v) < \mathsf{time}(u,t)\}} \geq \frac{t}{2}\}$, or

(2.10)
$$\Big\{ \sum_{v \in \mathcal{S} \setminus \mathcal{S}_i : \mathsf{depth}(v) < \frac{t}{2}} \sum_{t'=1}^{B} \mathbb{1}_{\{\mathsf{time}(v,t') < \mathsf{time}(u,t)\}} \geq \frac{t}{2} \Big\}$$

We would like to argue that we can further exclude from the sum all t' such that $t' > t$. Indeed, first consider the case $t' \geq 2 \cdot \mathsf{depth}(v)$. The algorithm can only play status (v,t') once t' is the lowest priority, which must happen after status (u,t) has the lowest priority. Hence $\mathsf{time}(v,t') < \mathsf{time}(u,t)$ is impossible and we can exclude status (v,t') from the sum. If $t' < 2 \cdot \mathsf{depth}(v)$, then $\mathsf{depth}(v) > \frac{t'}{2} > \frac{t}{2}$, and once again we can exclude status (v,t') from the sum. Thus

$$\begin{aligned} & (2.10) \\ \iff & \Big\{ \sum_{v \in \mathcal{S} \setminus \mathcal{S}_i : \mathsf{depth}(v) < \frac{t}{2}} \sum_{t'=1}^{t} \mathbb{1}_{\{\mathsf{time}(v,t') < \mathsf{time}(u,t)\}} \cdot \\ & \mathbb{1}_{\{\mathsf{time}(v) \leq t\}} \geq \frac{t}{2} \Big\} \\ \implies & \Big\{ \sum_{j \neq i} \sum_{v \in \mathcal{S}_j} \sum_{t'=1}^{t} \mathbb{1}_{\{\mathsf{time}(v,t') < \infty\}} \geq \frac{t}{2} \Big\} \end{aligned}$$

We can finally bound the quantity we wanted to.

$$\begin{aligned} & \Pr[\mathsf{time}(u,t) > t \mid \mathsf{time}(u,t) < \infty] \\ \leq \; & \Pr\Big[\sum_{j \neq i} \sum_{v \in \mathcal{S}_j} \sum_{t'=1}^{t} \mathbb{1}_{\{\mathsf{time}(v,t') < \infty\}} \geq \frac{t}{2} \\ & \mid \mathsf{time}(u,t) < \infty \Big] \\ = \; & \Pr\Big[\sum_{j \neq i} \sum_{v \in \mathcal{S}_j} \sum_{t'=1}^{t} \mathbb{1}_{\{\mathsf{time}(v,t') < \infty\}} \geq \frac{t}{2} \Big] \end{aligned}$$

where we remove the conditioning due to independence across arms. Now, let

$$Y_j = \min\Big\{ \sum_{v \in \mathcal{S}_j} \sum_{t'=1}^{t} \mathbb{1}_{\{\mathsf{time}(v,t') < \infty\}}, \frac{t}{2} \Big\}$$

for all $j \neq i$. The above probability is the same as $\Pr[\sum_{j \neq i} Y_j \geq \frac{t}{2}]$. Note that

$$\begin{aligned} \mathbb{E}\Big[\sum_{j \neq i} Y_j\Big] \; & \leq \; \sum_{j \neq i} \sum_{v \in \mathcal{S}_j} \sum_{t'=1}^{t} \Pr[\mathsf{time}(v,t') < \infty] \\ & \leq \; \sum_{t'=1}^{t} \sum_{v \in \mathcal{S}} \frac{x_{v,t'}}{3} \\ & \leq \; \frac{t}{3} \end{aligned}$$

where the third inequality follows from (2.2). We can do better than the Markov bound on $\Pr[\sum_{j \neq i} Y_j \geq \frac{t}{2}]$ because the random variables $\{Y_j\}_{j \neq i}$ are independent, and each Y_j is non-zero with probability at most $\frac{1}{3}$ (the algorithm plays arm j at all with probability $\sum_{t'=1}^{B} \frac{x_{\rho_{j,t'},t'}}{3}$, which is at most $\frac{1}{3}$ by combining (2.3), (2.5), (2.7)). So $\mathbb{E}[Y_j] \leq \frac{t}{6}$ for all $j \neq i$.

LEMMA 2.2. *Let $Y_1, \ldots, Y_m$ be independent non-negative random variables with individual expectations at most $\frac{t}{6}$ and sum of expectations at most $\frac{t}{3}$. Then $\Pr[\sum_{j=1}^{m} Y_j \geq \frac{t}{2}]$ is maximized when there are $m = 2$ random variables each of which take value $\frac{t}{2}$ with probability $\frac{1}{3}$, and value 0 otherwise. Therefore, $\Pr[\sum_{j=1}^{m} Y_j \geq \frac{t}{2}] \leq 1 - (1 - \frac{1}{3})^2 = \frac{5}{9}$.*

Lemma 2.2 would complete the proof that $\Pr[\mathsf{time}(u,t) > t \mid \mathsf{time}(u,t) < \infty] \geq \frac{5}{9}$ in Case 1, when $t \geq 2 \cdot \mathsf{depth}(u)$. The proof of Lemma 2.2 is written in the full version; it is a case analysis of probabilities. It uses the conjecture of Samuels in [20] for $n = 3$; the conjecture has been proven for $n \leq 4$ in [21] and recently addressed by Feige in [7]. The proof also uses a technical lemma of Bansal et al. in [3]; similar analyses appear in [6] and [3].

Case 2. Suppose $t < 2 \cdot \mathsf{depth}(u)$. Then $\mathsf{depth}(u)$ must be at least 1, so there must be some $v \in \mathsf{Par}(u)$ and $t' < t$ such that $\mathsf{time}(v,t') < \infty$, since $\mathsf{time}(u,t) < \infty$. Furthermore, the algorithm will play status (u,t) at time step $\mathsf{time}(v,t') + 1$, so $\mathsf{time}(u,t) \leq t$ will hold so long as $\mathsf{time}(v,t') \leq t'$ holds, since $t' < t$. Thus $\Pr[\mathsf{time}(u,t) \leq t \mid \mathsf{time}(u,t) < \infty] \geq \Pr[\mathsf{time}(v,t') \leq t \mid \mathsf{time}(v,t') < \infty]$, where we replaced $\mathsf{time}(u,t) < \infty$ with $\mathsf{time}(v,t') < \infty$ in the second probability since they

are equivalent when considering event $\mathsf{time}(v, t') \leq t$. We repeat this entire Case 2 procedure until $t' \geq 2 \cdot \mathsf{depth}(v)$, which allows us to use Case 1 to conclude $\Pr[\mathsf{time}(u,t) \leq t \mid \mathsf{time}(u,t) < \infty] \geq \frac{4}{9}$.

Therefore, the expected reward obtained by our algorithm is at least $\sum_{u \in \mathcal{S}} r_u \sum_{t=1}^{B} \frac{4}{27} x_{u,t}$, which is the same as $\frac{4}{27}\mathsf{OPT}_{\mathtt{LP}}$, completing the proof of Theorem 2.1.

2.3 Generalization to *MAB Superprocess with Multi-period Actions and Preemption*

Our algorithm views the arms as independent stochastic jobs and allocates the shared budget amongst them using rules that depend on the state of each arm (given a fixed LP solution). The same ideas still make sense when there are decisions at each node, and non-unit processing times. Some care must be taken to check that we can still write a polynomial LP relaxation, and that Lemma 2.1 still holds, but no new ideas are required, hence we leave the full proof to [16]. Another detail in the analysis is that in (2.10), we can only truncate the sum to B down to $\frac{3t}{2}$ (instead of t) when there are multi-period actions. This results in us requiring Samuels' conjecture for $n = 5$ (which is unproven), so we must use Markov instead, making the approximation factor much worse.

3 $(\frac{1}{2} - \varepsilon)$-approximation for Correlated Stochastic Knapsack with No Cancellation

3.1 Preliminaries *Correlated stochastic knapsack with no cancellation* is the following: there are n jobs, indexed by i. Each job has a stochastic processing time whose distribution is known beforehand. For convenience, we assume the processing time is always a positive integer. Each job also has a stochastic reward that could be correlated with its processing time. We are to schedule the jobs one by one, not being able to cancel a job in progress. The objective is to maximize, in expectation, the reward obtained from completed jobs before a time budget of B runs out.

Let R_i, S_i denote the random variables for the reward and processing time of job i, respectively. We define $\mathsf{ER}_{i,t}$ to be the expected reward obtained from job i when we start it at the beginning of time t. This is independent of any future decisions since jobs can't be cancelled, so $\mathsf{ER}_{i,t} = \sum_{t'=t+1}^{B+1} \mathbb{E}[R_i | S_i = t' - t] \cdot \Pr[S_i = t' - t]$.

3.2 Linear Programming Formulations We would like to write an LP relaxation, $(\mathtt{LP}')$, for this problem. Let $s_{i,t}$ be the probability job i has not been started before time t. Let $x_{i,t}$ be the probability job i is started at the beginning of time t.

$$\max \sum_{i=1}^{n} \sum_{t=1}^{B} \mathsf{ER}_{i,t} \cdot x_{i,t} \tag{3.11}$$

$$\sum_{i=1}^{n} \sum_{t'=1}^{t} x_{i,t'} \cdot \Pr[S_i > t - t'] \leq 1,\ t \in [B] \tag{3.12}$$

$$x_{i,t} \leq s_{i,t},\ i \in [n],\ t \in [B] \tag{3.13}$$

$$x_{i,t} \geq 0,\ i \in [n],\ t \in [B] \tag{3.14}$$

$$s_{i,1} = 1,\ i \in [n] \tag{3.15}$$

$$s_{i,t} = s_{i,t-1} - x_{i,t-1},\quad t > 1,\ i \in [n] \tag{3.16}$$

Our relaxation is tighter than that of [9]; we will prove its projection gap is 2 while we can find an example showing the projection gap of their LP is least 3.

Again, we assume the existence of some exponentially-large dynamic program which we refer to as $(\mathtt{DP}')$. The details, along with the proof that $\mathsf{OPT}_{\mathtt{DP}'} \leq \mathsf{OPT}_{\mathtt{LP}'}$, are written in the full version.

THEOREM 3.1. *Given a feasible solution to* $(\mathtt{LP}')$*, there exists a corresponding feasible solution to* $(\mathtt{DP}')$ *with all the probabilities scaled in half. Therefore, the projection gap of* $(\mathtt{LP}')$ *is at most* 2.

We first present a family of examples with $\frac{\mathsf{OPT}_{\mathtt{LP}'}}{\mathsf{OPT}_{\mathtt{DP}'}}$ approaching 2, providing a lower bound on the projection gap of $(\mathtt{LP}')$.

Let N be an arbitrary large integer. We have $n = 2$ jobs. Job 1 takes $N + 1$ time with probability $1 - \frac{1}{N}$, in which case it returns a reward of 1. It takes 1 time with probability $\frac{1}{N}$, in which case it returns no reward. Job 2 deterministically takes 1 time and returns a reward of 1. The budget is $B = N + 1$ time steps.

The optimal solution to $(\mathtt{LP}')$ will set $x_{1,1} = 1$ and $x_{2,2} = \ldots = x_{2,N+1} = \frac{1}{N}$, obtaining an objective value of $(1 - \frac{1}{N}) + N \cdot \frac{1}{N} = 2 - \frac{1}{N}$. However, any actual policy can never get more than 1 reward, since it cannot get reward from both jobs.

Let's analyze what goes wrong when we attempt to replicate the optimal solution to $(\mathtt{LP}')$ in an actual policy. Suppose we start job 1 at time 1 with probability $x_{1,1} = 1$. In the $\frac{1}{N}$ event it takes time 1, we start job 2 at time 2. The unconditional probability we start job 2 at time 2 is then $\frac{1}{N} = x_{2,2}$, as planned. However, in this event where job 1 took time 1, we cannot start job 2 again at time 3 (since it has already been processed at time 2), even though the LP is telling us to do so. The

LP fails to consider the fact that event "job 1 takes time 1" is *directly correlated* with event "job 2 is started at time 2", so we lose the reward for all of $x_{2,3}, \ldots, x_{2,N+1}$. Motivated by this example, we observe that if we only try to start job i at time t with probability $\frac{x_{i,t}}{2}$ instead, then we can obtain a solution to the large DP resembling a scaled copy of the LP solution.

Proof. [Proof of Theorem 3.1] Fix an optimal solution to $(\mathtt{LP}')$, $\{x_{i,t}, s_{i,t}\}$, and consider the following algorithm. It is in theory an admissible policy for the problem, even though the quantities $\mathsf{Free}(i,t)$ require exponential time to compute.

for $t = 1, \ldots, B$:
  if no job is currently being processed
    1) For all jobs i, let $\mathcal{F}_{i,t}$ be the event that in a run of this algorithm, there is no ongoing job at the beginning of time t and job i is available to be started. Let $\mathsf{Free}(i,t) = \Pr[\mathcal{F}_{i,t}]$.
    2) Start each remaining job i with probability $\frac{1}{\mathsf{Free}(i,t)} \cdot \frac{x_{i,t}}{2}$, and do nothing in this time step with probability $1 - \sum_i \frac{1}{\mathsf{Free}(i,t)} \cdot \frac{x_{i,t}}{2}$ (where the sum is over remaining jobs i).

We will induct on $t = 1, \ldots, B$ to prove that the sum of the probabilities $\frac{1}{\mathsf{Free}(i,t)} \cdot \frac{x_{i,t}}{2}$ in Step 2 is at most 1, ie. Step 2 is valid. It suffices to show that $\sum_{j=1}^n \frac{x_{j,t}}{2} \le \mathsf{Free}(i,t)$ for all $i \in [n]$. Let $\mathcal{X}_{j,t'}$ be the event that job j was started at the beginning of time t', for all $j \in [n]$, $t' < t$. So long as Step 2 was valid at time t', it is immediate that $\Pr[\mathcal{X}_{j,t'}] = \frac{x_{j,t'}}{2}$, so we can assume this in our induction hypothesis. Applying the union bound to the event $\overline{\mathcal{F}_{i,t}}$ (a job is currently being processed, or job i has already finished), we get

$$\begin{aligned}
& 1 - \mathsf{Free}(i,t) \\
\le & \sum_{t'<t} \Pr[\mathcal{X}_{i,t'}] + \sum_{j=1}^n \sum_{t'<t} \Pr[\mathcal{X}_{j,t'}] \cdot \Pr[S_j > t - t'] \\
= & \sum_{t'<t} \frac{x_{i,t'}}{2} + \sum_{j=1}^n \sum_{t'<t} \frac{x_{j,t'}}{2} \cdot \Pr[S_j > t - t'] \\
\le & \frac{1}{2} + \frac{1}{2}\big(1 - \sum_{j=1}^n x_{j,t}\big)
\end{aligned}$$

where the second inequality uses (3.12) and the fact that $\sum_{t'=1}^t x_{j,t'} \le 1 \ \forall j \in [n], t \in [B]$, which can be obtained by combining (3.13)-(3.16). This implies $\sum_{j=1}^n \frac{x_{j,t}}{2} \le \mathsf{Free}(i,t)$, completing the induction.

We have inductively proven that this policy is valid, and starts job j at time t' with probability $\frac{x_{j,t'}}{2}$ for all $j \in [n]$, $t' \in [B]$. Therefore, it corresponds to a feasible solution of $(\mathtt{DP}')$ obtaining an expected reward of $\frac{1}{2}\mathsf{OPT}_{\mathtt{LP}'}$, completing the proof that the projection gap of $(\mathtt{LP}')$ is at most 2.

THEOREM 3.2. *There is an* $(\mathtt{LP}')$*-based* $(\frac{1}{2} - \varepsilon)$*-approximation algorithm for correlated stochastic knapsack with no cancellation, with runtime polynomial in the input and* $\frac{1}{\varepsilon}$.

Proof. We make the previous policy polynomial-time by sampling the quantities $\mathsf{Free}(i,t)$. Fix some small $\varepsilon, \delta > 0$ that will be determined later. Define $\mu_{\varepsilon,\delta}$ to be the constant $\frac{3\ln(2\delta^{-1})}{\varepsilon^2}$. Consider the following subroutine, which takes in the quantities as parameters, up to some time step t:

$\mathsf{Policy}\ (t,\ \{f(i,t')\}_{i\in[n],t'\in[t]})$
for $t' = 1, \ldots, t$:
  if no job is currently being processed
    Start each remaining job i with probability $\frac{1}{f(i,t')} \cdot \frac{x_{i,t'}}{2} \cdot (1-\varepsilon)^2$; skip the time step otherwise.

Now, consider the following algorithm that runs simulations and records the sample averages as $\mathsf{Free}^{\mathsf{emp}}(i,t)$. After it finishes, we can run Policy once in reality, with these parameters, to claim the desired expectation.

for $t = 1, \ldots, B$:
  1) Simulate $\mathsf{Policy}\ (t-1,\ \{\mathsf{Free}^{\mathsf{emp}}(i,t')\}_{i\in[n],t'\in[t-1]})$ a total of $M = \frac{8Bn}{\varepsilon} \cdot \mu_{\varepsilon,\delta}$ times. As before, let $\mathcal{F}_{i,t}$ be the event that job i is available to be started and there is no ongoing job at the beginning of time t (at the end of the simulation); note that this depends on the recorded $\{\mathsf{Free}^{\mathsf{emp}}(i,t')\}_{i\in[n],t'\in[t-1]}$. Let $C_{i,t}$ count the number of occurences of the event $\mathcal{F}_{i,t}$.
  2) For all i, if $C_{i,t} > \mu_{\varepsilon,\delta}$, then set $\mathsf{Free}^{\mathsf{emp}}(i,t) = \frac{C_{i,t}}{M}$; otherwise set $\mathsf{Free}^{\mathsf{emp}}(i,t) = \sum_{j=1}^n \frac{x_{j,t}}{2}$.

Also as before, define $\mathsf{Free}(i,t) = \Pr[\mathcal{F}_{i,t}]$, and let $\mathcal{X}_{i,t}$ be the event job i is started at time t when Policy is ran with the recorded parameters $\{\mathsf{Free}^{\mathsf{emp}}(i,t')\}_{i\in[n],t'\in[t]}$. We use the fact from sampling that so long as $C_{i,t} > \mu_{\varepsilon,\delta}$, $\frac{C_{i,t}}{M}$ is within $(1 \pm \varepsilon)$ of $\mathsf{Free}(i,t)$ with probability at least $(1 - \delta)$ [18]. Furthermore, if $C_{i,t} \le \mu_{\varepsilon,\delta}$, then with probability at least $(1 - O(\delta))$, it must have been that $\mathsf{Free}(i,t) \le \frac{\varepsilon}{4Bn}$. Indeed, if to the contrary $\mathsf{Free}(i,t) > \frac{\varepsilon}{4Bn}$, then $\mathbb{E}[C_{i,t}] > 2\mu_{\varepsilon,\delta}$, so by Chernoff bound, $\Pr[C_{i,t} \le \mu_{\varepsilon,\delta}] = O(\delta^{\frac{1}{\varepsilon^2}})$. We call these two $O(\delta)$ events *failures* that can happen for some $i \in [n]$, $t \in [B]$.

The probability of any failure at some point in the algorithm is $Bn(\delta + O(\delta)) = O(Bn\delta)$, by union bound. Assuming zero failures, we will inductively prove that:

- $\frac{(1-\varepsilon)^2}{1+\varepsilon}\cdot\frac{x_{i,t}}{2} \le \Pr[\mathcal{X}_{i,t}] \le \max\{(1-\varepsilon)\cdot\frac{x_{i,t}}{2}, \frac{\varepsilon}{4Bn}\}$, for all $i \in [n]$
- $\sum_{i=1}^{n} \frac{1}{\mathsf{Free}^{\mathsf{emp}}(i,t)} \cdot \frac{x_{i,t}}{2} \cdot (1-\varepsilon)^2 \le 1$ (sum of probabilities in Policy is at most 1)

Using a similar analysis as before, and the inductive hypothesis that $\Pr[\mathcal{X}_{j,t'}] \le (1-\varepsilon)\cdot\frac{x_{j,t'}}{2} + \frac{\varepsilon}{4Bn}$ for all $j \in [n]$, $t' < t$, we get

$$\begin{aligned}
& 1 - \mathsf{Free}(i,t) \\
\le\ & \sum_{t'<t} \big((1-\varepsilon)\cdot\frac{x_{i,t'}}{2} + \frac{\varepsilon}{4Bn}\big) + \\
& \sum_{j=1}^{n}\sum_{t'<t} \big((1-\varepsilon)\cdot\frac{x_{j,t'}}{2} + \frac{\varepsilon}{4Bn}\big)\cdot\Pr[S_j > t-t'] \\
\le\ & (1-\varepsilon)\sum_{t'<t}\frac{x_{i,t'}}{2} + \frac{\varepsilon}{4n} + \\
& \sum_{j=1}^{n}\sum_{t'<t}\frac{x_{j,t'}}{2}\cdot\Pr[S_j > t-t'] + \frac{\varepsilon}{4} \\
\le\ & (1-\varepsilon)\cdot\frac{1}{2} + \frac{\varepsilon}{4} + \frac{1}{2}\big(1 - \sum_{j=1}^{n} x_{j,t}\big) + \frac{\varepsilon}{4}
\end{aligned}$$

which proves that $\mathsf{Free}(i,t) \ge \sum_{j=1}^{n}\frac{x_{j,t}}{2}$ for all $i \in [n]$, a key fact. Now we prove both bullets:

- By the description of the algorithm, $\Pr[\mathcal{X}_{i,t}] = \mathsf{Free}(i,t)\cdot\frac{1}{\mathsf{Free}^{\mathsf{emp}}(i,t)}\cdot\frac{x_{i,t}}{2}\cdot(1-\varepsilon)^2$. If $(1-\varepsilon) \le \frac{\mathsf{Free}^{\mathsf{emp}}(i,t)}{\mathsf{Free}(i,t)} \le (1+\varepsilon)$, then we get $\frac{(1-\varepsilon)^2}{1+\varepsilon}\cdot\frac{x_{i,t}}{2} \le \Pr[\mathcal{X}_{i,t}] \le (1-\varepsilon)\cdot\frac{x_{i,t}}{2}$, and are done. Otherwise, assuming no failures, $C_{i,t} \le \mu_{\varepsilon,\delta}$ and $\mathsf{Free}(i,t) \le \frac{\varepsilon}{4Bn}$. Also, $\mathsf{Free}^{\mathsf{emp}}(i,t)$ will get set to $\sum_{j=1}^{n}\frac{x_{j,t}}{2}$, so $\frac{1}{\mathsf{Free}^{\mathsf{emp}}(i,t)}\cdot\frac{x_{i,t}}{2} \le 1$ and thus $\Pr[\mathcal{X}_{i,t}] \le \frac{\varepsilon}{4Bn}\cdot(1-\varepsilon)^2$, which suffices to prove the upper bound. For the lower bound, the key fact tells us $\mathsf{Free}(i,t) \ge \mathsf{Free}^{\mathsf{emp}}(i,t)$, so $\Pr[\mathcal{X}_{i,t}] \ge \frac{x_{i,t}}{2}\cdot(1-\varepsilon)^2$, completing the proof.
- Assuming no failures, $\mathsf{Free}^{\mathsf{emp}}(i,t)$ will either get set to $\sum_{j=1}^{n}\frac{x_{j,t}}{2}$, or be at least $(1-\varepsilon)\cdot\mathsf{Free}(i,t)$, which is at least $(1-\varepsilon)\cdot\sum_{j=1}^{n}\frac{x_{j,t}}{2}$, by the key fact. Either way, $\mathsf{Free}^{\mathsf{emp}}(i,t)$ is at least $(1-\varepsilon)\cdot\sum_{j=1}^{n}\frac{x_{j,t}}{2}$, so the desired sum is at most $(1-\varepsilon)$, completing the proof.

We get an algorithm that obtains at least $\frac{(1-\varepsilon)^2}{1+\varepsilon}\cdot\frac{1}{2}$ of the optimum reward in expectation, fails with probability $O(Bn\delta)$, and runs in time polynomial in the input, $\frac{1}{\varepsilon}$, and $\ln(\frac{1}{\delta})$. Treating a failed run as a run obtaining 0 reward, we can choose $\delta = \frac{\varepsilon}{Bn}$ to complete the proof of Theorem 3.1.

3.3 Generalization to *MAB Superprocess with Multi-period Actions and No Preemption*

In the algorithm above, the arms are independent stochastic jobs that cannot be cancelled, and our algorithm guarantees to start each job i at each time t with probability $\frac{x_{i,t}}{2}$ (given a fixed LP solution). If the arms had the more complicated structure of MAB superprocesses with multi-period actions, we can still think of the instructions given by the LP variables as scheduling independent stochastic jobs that cannot be cancelled. Note that if the LP tells us to cancel a job, so long as it won't tell us to start it again later, we can just treat this as the job terminating early. This fits under the algorithm above, except the distribution of a job is no longer invariant from the time t it starts. However, nothing in the above algorithm required this invariant; it only required that a job cannot be started twice. Some care must be taken to check that we can still write a polynomial LP relaxation, and that the sampling still works, but no new ideas are required for the generalization, hence we leave the full proof to [16].

References

[1] D. P. Bertsekas, *Dynamic programming and optimal control*, vol. 1, Athena Scientific Belmont, 1995.

[2] A. Bhalgat, A. Goel, S. Khanna, *Improved approximation results for stochastic knapsack problems*, SODA, 2011, pp. 1647–1665.

[3] N. Bansal, A. Gupta, J. Li, J. Mestre, V. Nagarajan, A. Rudra, *When LP is the cure for your matching woes: improved bounds for stochastic matchings*, Algorithms-ESA, 2010, pp. 218–229.

[4] A. Bhalgat, *A $(2+\varepsilon)$-approximation algorithm for the stochastic knapsack problem*, available online at http://www.cis.upenn.edu/~bhalgat/2-approx-stochastic.pdf, 2011.

[5] R. L. Carraway, R. L. Schmidt, L. R. Weatherford, *An algorithm for maximizing target achievement in the stochastic knapsack problem with normal returns*, Naval Research Logistics **40-2** (1993), pp. 161–173.

[6] B. C. Dean, M. X. Goemans, J. Vondrák, *Approximating the stochastic knapsack problem: the benefit of adaptivity*, Math. Oper. Res. **33-4** (2008), pp. 945–964.

[7] U. Feige, *On sums of independent random variables with unbounded variance and estimating the average degree in a graph*, SIAM J. Comput. **35-4** (2006), pp. 964–984 (electronic).

[8] J. Gittins, K. Glazebrook, R. Weber, *Multi-armed bandit allocation indices*, Wiley Online Library, 1989.

[9] A. Gupta, R. Krishnaswamy, M. Molinaro, R. Ravi, *Approximation algorithms for correlated knapsacks and non-martingale bandits*, FOCS, 2011, pp. 827–836.

[10] A. Gupta, R. Krishnaswamy, V. Nagarajan, R. Ravi, *Approximation algorithms for stochastic orienteering*, SODA, 2012, pp. 1522–1538.
[11] S. Guha, K. Munagala, *Sequential design of experiments via linear programming*, arXiv:0805.2630v2, 2008.
[12] S. Guha, K. Munagala, *Approximation algorithms for bayesian multi-armed bandit problems*, arXiv:1306.3525, 2013.
[13] T. İlhan, S. Iravani, M. S. Daskin, *The adaptive knapsack problem with stochastic rewards*, Oper. Res. **59-1** (2011), pp. 242–248.
[14] J. M. Kessler, *United States air force fighter jet maintenance models: effectiveness of index policies*, Master's Thesis, MIT Oper. Res. Center, 2013.
[15] J. Li, W. Yuan, *Stochastic combinatorial optimization via Poisson approximation*, STOC, 2013, pp. 971–980.
[16] W. Ma, *Improvements and generalizations of stochastic knapsack and multi-armed bandit algorithms: full version*, arXiv:1306.1149, 2013.
[17] A. Mehta, D. Panigrahi, *Online matching with stochastic rewards*, FOCS, 2012, pp. 728-737.
[18] R. Motwani, P. Raghavan, *Randomized algorithms*, Cambridge university press, 1995.
[19] M. Pinedo, *Scheduling: theory, algorithms, and systems*, Springer Science+ Business Media, 2012.
[20] S. M. Samuels, *On a Chebyshev-type inequality for sums of independent random variables*, Ann. Math. Stat. **37-1** (1966), pp. 248-259.
[21] S. M. Samuels, *More on a Chebyshev-type inequality for sums of independent random variables*, Defense Technical Information Center, 1968.