Bandits with Knapsacks: Dynamic procurement for crowdsourcing*

Ashwinkumar Badanidiyuru † Robert Kleinberg ‡ Aleksandrs Slivkins §

Abstract

In a basic version of the *dynamic procurement* problem, the algorithm has a budget B to spend, and is facing n agents (potential sellers) that are arriving sequentially. The algorithm offers a take-it-or-leave-it price to each arriving seller; the sellers value for an item is an independent sample from some fixed (but unknown) distribution. The goal is to maximize the number of items bought. This problem is particularly relevant to the emerging domain of *crowdsourcing*, where agents correspond to the (relatively inexpensive) workers on a crowdsourcing platform such as Amazon Mechanical Turk, and "items" bought/sold correspond to simple jobs ("microtasks") that can be performed by these workers. The algorithm corresponds to the "client": an entity that submits jobs and benefits from them being completed. The basic formulation admits various generalizations, e.g. to multiple job types. We also address an alternative model in which the requester posts offers to the entire crowd.

We model the dynamic procurement problems as *multi-armed bandit* problems with a budget constraint. We define "bandits with knapsacks": a broad class of multi-armed bandit problems with knapsack-style resource-utilization constraints which subsumes dynamic procurement and a host of other applications. A distinctive feature of our problem, in comparison to the existing regret-minimization literature, is that the optimal policy for a given latent distribution may significantly outperform the policy that plays the optimal fixed arm. Consequently, achieving sublinear regret in the bandits-with-knapsacks problem is significantly more challenging than in conventional bandit problems.

Our main result is an algorithm for a version of bandits-with-knapsacks with finitely many possible actions. It is a primal-dual algorithm with multiplicative updates; the regret of this algorithm is close to the information-theoretic optimum. We derive corollaries for dynamic procurement using uniform discretization of prices.

1 Introduction

Dynamic procurement. The *dynamic procurement* problem is similar to *dynamic pricing* problems studied in the literature, except that now an algorithm is buying rather than selling.

The basic problem formulation (*unit supply*) is as follows. The algorithm has a budget B to spend, and is facing n agents (potential sellers) that are arriving sequentially. Each seller is interested in selling one item. Each sellers value for an item is an independent sample from some fixed (but unknown) distribution with support $[0, 1]$. The algorithm

*This is a re-focused and shortened version of [11, 12], to appear at *IEEE FOCS 2013*. That paper, titled "Bandits with Knapsacks", is on the general problem, and discusses dynamic procurement among other applications. Compared to the present version, it includes an alternative algorithm based on very different techniques, a fleshed out discussion of applications beyond dynamic procurement, and a matching lower bound (in the worst-case over the general problem).

†Department of Computer Science, Cornell University, Ithaca NY, USA. Email: ashwinkumarbv@gmail.com.

‡Department of Computer Science, Cornell University, Ithaca NY, USA. Email: rdk@cs.cornell.edu.

§Microsoft Research Silicon Valley, Mountain View CA, USA. Email: slivkins@microsoft.com.

offers a take-it-or-leave-it price to each arriving agent. The goal is to maximize the number of items bought. This problem has been studied in [10]; they achieve a multiplicative constant-factor approximation with respect to the optimal dynamic policy, with a very large constant ($\sim 10^9$).

The problem is particularly relevant to the emerging domain of *crowdsourcing*, where agents correspond to the (relatively inexpensive) workers on a crowdsourcing platform such as Amazon Mechanical Turk, and "items" bought/sold correspond to simple jobs ("microtasks") that can be performed by these workers. The algorithm corresponds to the "requester": an entity that submits jobs and benefits from them being completed. The (basic) dynamic procurement model captures an important issue in crowdsourcing that a requester interacts with multiple users with unknown values-per-item, and can adjust its behavior (such as the posted price) over time as it learns the distribution of users. While this basic model ignores some realistic features of crowdsourcing environments, some of these limitations are addressed by the generalizations which we present below. In particular, our results apply to an alternative model in which the requester posts offers to the entire crowd.

Dynamic procurement as a multi-armed bandit problem. For more than fifty years, the multi-armed bandit problem (henceforth, *MAB*) has been the predominant theoretical model for sequential decision problems that embody the tension between exploration and exploitation, "the conflict between taking actions which yield immediate reward and taking actions whose benefit (e.g. acquiring information or preparing the ground) will come only later," to quote Whittle's apt summary [39]. Owing to the universal nature of this conflict, it is not surprising that MAB algorithms have found diverse applications ranging from medical trials, to communication networks, to Web search and advertising.

Let us cast dynamic procurement as an MAB-style problem. Agents correspond to time rounds, and n is the time horizon. Arms correspond to feasible prices. The "reward" of an algorithm is simply the total number of jobs/items that are procured. For each price p, letting $S(p)$ be the probability of procuring a job/item at price p, the expected reward is $S(p)$. However, unlike the traditional MAB formulations, dynamic procurement has a budget constraint; the expected budget consumption for price p is $pS(p)$.

The general problem: bandits with knapsacks. We observe that it is a common feature in many of the application domains for MAB that there are some limited-supply resources that are consumed during the decision process. For example, scientists experimenting with alternative medical treatments may be limited not only by the number of patients participating in the study but also by the cost of materials used in the treatments. A website experimenting with displaying advertisements is constrained not only by the number of users who visit the site but by the advertisers' budgets. A retailer engaging in price experimentation faces inventory limits along with a limited number of consumers. The literature on MAB problems lacks a general model that encompasses these sorts of decision problems with supply limits. Our paper contributes such a model, and it presents algorithms whose regret (normalized by the payoff of the optimal policy) converges to zero as the resource budget and the optimal payoff tend to infinity.

Our problem formulation, which we call the *bandits with knapsacks* problem (henceforth, `BwK`), is easy to state. A learner has a fixed set of potential actions, denoted by X and known as *arms*. (In our main results, X will be finite; we extend to infinite set of arms using dicretization.) Over a sequence of time steps, the learner chooses an arm and observes two things: a *reward* and a *resource consumption vector*. Rewards are scalar-valued whereas resource consumption vectors are d-dimensional. For each resource there is a pre-specified *budget* representing the maximum amount that may be consumed, in total. At the first time τ when the total consumption of some resource exceeds its budget, the process stops.

The conventional MAB problem with a finite time horizon, naturally fits into this framework: there is a single resource called "time", one unit of which is deterministically consumed in each decision period.

For the basic dynamic procurement problem, there are two resources: time and money. If price p is offered and accepted, the reward is 1 and the resource consumption is $\left[\begin{smallmatrix} p \\ 1 \end{smallmatrix}\right]$. If the offer is declined, the reward is 0 and the resource consumption is $\left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right]$.

A host of other applications is described in the full version [12], most notably applications to dynamic pricing and to dynamic allocation of pay-per-click ads. Other applications include repeated auctions, network routing, and scheduling. We omit a detailed discussion of other applications of `BwK` from this version.

Benchmark and regret. We will assume that the data for a fixed arm x in each time step (i.e. the reward and resource consumption vector) are i.i.d. samples from a fixed joint distribution on $[0,1] \times [0,1]^d$, called the *latent distribution* for arm x. The performance of an algorithm will be measured by its *regret*: the worst case, over all possible tuples of latent distributions, of the difference between the algorithm's expected reward and the expected reward of the benchmark:

an optimal policy given foreknowledge of the latent distribution. In a conventional MAB problem, the optimal policy given foreknowledge of the latent distribution is to play a fixed arm, namely the one with the highest expected reward. In the BwK problem, the optimal policy for a given distribution is more complex: the choice of optimal arm depends on the remaining supply of each resource. In fact, we doubt there is a polynomial-time algorithm to compute the optimal policy; similar problem in optimal control have long been known to be PSPACE-hard [34].

Nevertheless we are able to bound the regret of our algorithms with respect to the optimal policy, by showing that the reward of the optimal policy is closely approximated by that of the best *time-invariant mixture* of arms, i.e. a policy that samples in each period from a fixed probability distribution over arms regardless of the remaining resource supplies, and that maximizes expected reward subject to this constraint. The fact that a mixture of arms may be strictly superior to any fixed arm (see Appendix A) highlights a qualitative difference between the BwK problem and conventional MAB problems, and it illustrates one reason why the former problem is significantly more difficult to solve. In fact, we are not aware of any published work on explore-exploit problems in which an algorithm significantly improves over the best-fixed-arm benchmark.

Our results and techniques: bandits with knapsacks. In analyzing MAB algorithms one typically expresses the regret as a function of the time horizon, T. The regret guarantee is considered nontrivial if this function grows sublinearly as $T \to \infty$. In the BwK problem a regret guarantee of the form $o(T)$ may be unacceptably weak because supply limits prevent the optimal policy from achieving a reward close to T. An illustrative example is the dynamic pricing problem with supply $k \ll T$: the seller can only sell k items, each at a price of at most 1, so bounding the regret by any number greater than k is worthless.

We instead seek regret bounds that are sublinear in OPT, the expected reward of the optimal policy, or (at least) sublinear in M_{LP}, the maximal possible value of OPT given the budget constraints. To achieve sublinear regret, the algorithm must be able to explore each arm a significant number of times without exhausting its resource budget. Accordingly we also assume, for some $B \geq 1$, that the amount of any resource consumed in a single round is guaranteed to be no more than $1/B$ fraction of that resource's budget, and we parameterize our regret bound by B. We present an algorithm (called PD-BwK) whose regret is sublinear in OPT as both OPT and B tend to infinity. More precisely, denoting the number of arms by m, our algorithm's regret is

$$\widetilde{O}\left(\sqrt{m\,\texttt{OPT}} + \texttt{OPT}\sqrt{m/B}\ \right). \tag{1}$$

Algorithm PD-BwK is a primal-dual algorithm based on the multiplicative weights update method. It maintains a vector of "resource costs" that is adjusted using multiplicative updates. In every period it estimates each arm's expected reward and expected resource consumption, using upper confidence bounds for the former and lower confidence bounds for the latter; then it plays the most "cost-effective" arm, namely the one with the highest ratio of estimated resource consumption to estimated resource cost, using the current cost vector. Although confidence bounds and multiplicative updates are the bread and butter of online learning theory, we consider this way of combining the two techniques to be quite novel. In particular, previous multiplicative-update algorithms in online learning theory — such as the Exp3 algorithm for MAB [7] or the weighted majority [31] and Hedge [20] algorithms for learning from expert advice — applied multiplicative updates to the probabilities of choosing different arms (or experts). Our application of multiplicative updates to the dual variables of the LP relaxation of BwK is conceptually quite a different usage of this technique.

1.1 Our results: dynamic procurement

In dynamic procurement (as in some other applications of BwK) the action space X is infinite, so the algorithms for finite action sets are not immediately applicable. However, X has some structure that we can leverage. We use *uniform discretization*: we select a subset $S \subset X$ of arms which are, in some sense, uniformly spaced in X, and apply a BwK algorithm for this subset.

This generic approach has been successfully used in past work on MAB on metric spaces (e.g., [25, 24, 29, 32]) and dynamic pricing (e.g., [27, 14, 13, 9]) to provide worst-case optimal regret bounds. When $X = [0, 1]$, a typical choice for S is $\epsilon\,\mathbb{N} \cap [0, 1]$; we call it the *additive ϵ-mesh*. To implement this approach in the setting of BwK, we need to define an appropriate notion of discretization (which abstracts away the useful properties of the additive ϵ-mesh in prior work), and argue that it does not cause too much damage. Compared to the usage of discretization in prior work, we need to deal with two technicalities. First, we need to argue about distributions over arms rather than individual arms. Second, in order to compare an arm with its "image" in the discretization, we need to consider the difference in the ratio of expected reward to expected consumption, rather than the difference in rewards.

Unit supply. For dynamic procurement with unit supply, we find that arbitrarily small prices are not amenable to discretization. Instead, we focus on prices $p \geq p_0$, where $p_0 \in (0,1)$ is a parameter to be adjusted, and construct an ϵ-discretization on the set $[p_0, 1]$. We find that the additive δ-mesh (for a suitably chosen δ) is not the most efficient way to construct an ϵ-discretization in this domain. Instead, we use a mesh of the form $\{\frac{1}{1+j\epsilon} : j \in \mathbb{N}\}$; we call it the *hyperbolic ϵ-mesh*. Then we obtain an ϵ-discretization with a significantly less arms. Optimizing the parameters p_0 and ϵ, we obtain regret $\tilde{O}\left(n/B^{1/4}\right)$.

Note that we obtain sublinear regret if and only if OPT is small compared to $n/B^{1/4}$. This greatly improves over the (large) constant factor approximation in [10].

Extension: non-unit supply. We consider an extension where each agent may be interested in more than one item. For example, a worker may be interested in performing several jobs. In each round t, the algorithm offers to buy up to λ_t units at a fixed price p_t per unit, where the pair (p_t, λ_t) is chosen by the algorithm. The t-th agent then chooses how many units to sell. Agents' valuations may be non-linear in k_t, the number of units they sell. Each agent chooses k_t that maximizes her utility. We restrict $\lambda_t \leq \Lambda$, where Λ is an a-priori chosen parameter.

We model it as a BwK domain with a single resource (money) and action space $X = [0,1] \times \{0, 1, \ldots, \Lambda\}$. Note that for each arm $x = (p_t, \lambda_t)$ the expected reward is $r(x, \mu) = \mathbb{E}[k_t]$, and the expected consumption is $c(x, \mu) = p_t\ \mathbb{E}[k_t]$. As in the basic dynamic procurement problem, we focus on prices $p \geq p_0$ and use the hyperbolic ϵ-mesh on $[p_0, 1]$ for discretization, for some parameters $p_0, \epsilon \in (0,1)$. Optimizing the ϵ, we obtain regret $\tilde{O}(\Lambda^{3/2} n/B^{1/4})$. In a more restricted version with $\lambda_t = \Lambda$, we obtain regret $\tilde{O}(\Lambda^{5/4} n/B^{1/4})$.

Alternative interpretation: offers to the "crowd". In current crowdsourcing markets, requesters may be required to post their job offers to the entire "crowd" of workers, rather than target individual workers. This setting is easily captured by our model: in each round t, an algorithm picks a pair (p_t, λ_t), which means that it offers up to λ_t jobs for a fixed price p_t per job, and the outcome is that k_t jobs are picked up. We assume that the k_t is an independent sample from some distribution that depends only on the pair (p_t, λ_t), but not on t. As before, the requester has a budget B, and wishes to maximize the total number of completed jobs. Formally, this is exactly the same model as dynamic procurement with non-unit supply, so our results apply.

The independence assumption on k_t is crucial to the model. Informally, this assumption is two-fold: the workers are myopic: they only take into account their utility in the current round, and the "crowd" is stationary: its relevant properties do not change over time. With this assumption in place, the details of workers' utilities, decision making, and possible collusion, are irrelevant to the algorithm.

Other extensions. Further, we can model and handle a number of other extensions. As in "other extensions" for dynamic pricing, will assume that the action space is restricted to a specific finite subset $S \subset X$ that is given to the algorithm, and bound the *S-regret* – regret with respect to the restricted action space.

• *Multiple types of jobs*. There are ℓ types of jobs requested on the crowdsourcing platform. Each agent t has a private cost $v_{t,i} \in [0,1]$ for each type i; the vector of private costs comes from a fixed but unknown distribution (i.e., arbitrary correlations are allowed). The algorithm derives utility $u_i \in [0,1]$ from each job of type i. In each round t, the algorithm offers a vector of prices $(p_{t,1}, \ldots, p_{t,\ell})$, where $p_{t,i}$ is the price for one job of type i. For each type i, the agent performs one job of this type if and only if $p_{t,i} \geq v_{t,i}$, and receives payment $p_{t,i}$ from the algorithm.

Here arms correspond to the ℓ-dimensional vectors of prices, so that the (full) action space is $X = [0,1]^\ell$. Given the restricted action space $S \subset X$, we obtain S-regret $\tilde{O}(\ell)(\sqrt{n\,|S|} + n\sqrt{\ell\,|S|/B})$.

• *Additional features.* We can also model more complicated "menus" so that each agent can perform several jobs of the same type. Then in each round, for each type i, the algorithm specifies the maximal offered number of jobs of this type and the price per one such job.

We can also incorporate constraints on the maximal number of jobs of each type that is needed by the requester, and/or the maximal amount of money spend on each type.

• *Competitive environment.* There may be other requesters in the system, each offering its own vector of prices in each round. (This is a realistic scenario in crowdsourcing, for example.) Each seller / worker chooses the requester and the price that maximize her utility. One standard way to model such competitive environment is to assume that the "best offer" from the competitors is a vector of prices which comes from a fixed but unknown distribution. This

can be modeled as a BwK instance with a different distribution over outcomes which reflects the combined effects of the demand distribution of agents and the "best offer" distribution of the environment.

1.2 Related work

The study of prior-free algorithms for stochastic MAB problems was initiated by Lai and Robbins [30], who provided an algorithm whose regret after T time steps grows asymptotically as $O(\log T)$, and showed that this bound was tight up to constant factors. Their analysis was only asymptotic and did not provide an explicit regret bound that holds for finite T, a shortcoming that was overcome by the UCB1 algorithm [6]. Subsequent work supplied algorithms for stochastic MAB problems in which the set of arms is infinite and the payoff function is linear [1, 16], concave [2], or Lipschitz-continuous [3, 8, 15, 25, 29, 28]. Confidence bound techniques have been an integral part of all these works, and they remain integral to ours.

As explained earlier, stochastic MAB problems constitute a very special case of bandits with knapsacks, in which there is only one type of resource and it is consumed deterministically at rate 1. Several papers have considered the natural generalization in which there is a single resource, with deterministic consumption, but different arms consume the resource at different rates. Guha and Munagala [22] gave a constant-factor approximation algorithm for the Bayesian case of this problem, which was later generalized by Gupta et al. [23] to settings in which the arms' reward processes need not be martingales. Tran-Thanh et al. [36, 37, 38] presented prior-free algorithms for this problem; the best such algorithm achieves a regret guarantee qualitatively similar to that of the UCB1 algorithm.

Two recent papers introduced dynamic pricing [9] and procurement [10] problems which, in hindsight, can be cast as special cases of BwK featuring a two-dimensional resource constraint. The dynamic pricing paper presents an algorithm with $\widetilde{O}(k^{2/3})$ regret with respect to the best single price, when the supply is k. As explained in the full version, we strengthen this result by achieving the same regret bound with respect to the optimal dynamic policy, which is a much stricter benchmark in some cases. The main result of the dynamic procurement paper is a posted-price algorithm that achieves a constant-factor approximation to the optimum, albeit with a prohibitively large constant (at least in the tens of thousands). A corollary of our main result is a new posted-price algorithm for dynamic procurement whose approximation factor approaches 1 as the ratio between the budget and the maximum cost of procuring a single service tends to infinity.

While BwK is primarily an online learning problem, it also has elements of a stochastic packing problem. The literature on prior-free algorithms for stochastic packing has flourished in recent years, starting with prior-free algorithms for the stochastic AdWords problem [17], and continuing with a series of papers extending these results from AdWords to more general stochastic packing integer programs while also achieving stronger performance guarantees [4, 18, 19, 33]. A running theme of these papers (and also of the primal-dual algorithm in this paper) is the idea of estimating of an optimal dual vector from samples, then using this dual to guide subsequent primal decisions. Particularly relevant to our work is the algorithm of [18], in which the dual vector is adjusted using multiplicative updates, as we do in our algorithm. However, unlike the BwK problem, the stochastic packing problems considered in prior work are not learning problems: they are full information problems in which the costs and rewards of decisions in the past and present are fully known. (The only uncertainty is about the future.) As such, designing algorithms for BwK requires a substantial departure from past work on stochastic packing. Our primal-dual algorithm depends upon a hybrid of confidence-bound techniques from online learning and primal-dual techniques from the literature on solving packing LPs; combining them requires entirely new techniques for bounding the magnitude of the error terms that arise in the analysis.

2 Preliminaries

BwK: problem formulation. There is a fixed and known, finite set of m *arms* (possible actions), denoted X. There are d resources being consumed. In each round t, an algorithm picks an arm $x_t \in X$, receives reward $r_t \in [0,1]$, and consumes some amount $c_{t,i} \in [0,1]$ of each resource i. The values r_t and $c_{t,i}$ are revealed to the algorithm after the round. There is a hard constraint $B_i \in \mathbb{R}_+$ on the consumption of each resource i; we call it a *budget* for resource i. The algorithm stops at the earliest time τ when one or more budget constraint is violated; its total reward is equal to the sum of the rewards in all rounds strictly preceding τ. The goal of the algorithm is to maximize the expected total reward.

The vector $(r_t; c_{t,1}, c_{t,2}, \ \ldots\ , c_{t,d}) \in [0,1]^{d+1}$ is called the *outcome vector* for round t. We assume *stochastic outcomes*: if an algorithm picks arm x, the outcome vector is chosen independently from some fixed distribution π_x over $[0,1]^{d+1}$. The distributions π_x, $x \in X$ are latent. The tuple $(\pi_x : x \in X)$ comprises all latent information in the

problem instance. A particular BwK setting (such as "dynamic pricing with limited supply") is defined by the set of all feasible tuples $(\pi_x : x \in X)$. This set, called the *BwK domain*, is known to the algorithm.

We will assume that there is a fixed *time horizon* T, known in advance to the algorithm, such that the process is guaranteed to stop after at most T rounds. One way of assuring this is to assume that there is a specific resource, say resource 1, such that every arm deterministically consumes B_1/T units whenever it is picked. We make this assumption henceforth. Without loss of generality, $B_i \leq T$ for every resource i.

For technical convenience, we assume there exists a *null arm*: an arm with 0 reward and 0 consumption. Equivalently, an algorithm is allowed to spend a unit of time without doing anything.

Benchmark. We compare the performance of our algorithms to the expected total reward of the optimal dynamic policy given all the latent information, which we denote by OPT. Note that OPT depends on the latent structure μ, and therefore is a latent quantity itself. Time-invariant policies — those which use the same distribution $\mathcal{D}$ over arms in all rounds — will also be relevant to the analysis of one of our algorithms. Let $\texttt{REW}(\mathcal{D}, \mu)$ denote the expected total reward of the time-invariant policy that uses distribution $\mathcal{D}$.

Uniform budgets. We say that the budgets are *uniform* if $B_i = B$ for each resource i. Any BwK instance can be reduced to one with uniform budgets by dividing all consumption values for every resource i by B_i/B, where $B = \min_i B_i$. (That is tantamount to changing the units in which we measure consumption of resource i.) Our technical results are for BwK with uniform budgets. We will assume uniform budgets B from here on.

Useful notation. Let $\mu_x = \mathbb{E}[\pi_x] \in [0,1]^{d+1}$ be the expected outcome vector for each arm x, and denote $\mu = (\mu_x : x \in X)$. We call μ the *latent structure* of a problem instance.

For notational convenience, we will write $\mu_x = (\ r(x,\mu);\ c_1(x,\mu)\ ,\ \ldots\ , c_d(x,\mu)\ )$. Also, we will write the expected consumption as a vector $c(x,\mu) = (\ c_1(x,\mu)\ ,\ \ldots\ , c_d(x,\mu)\ )$.

When discussing our primal-dual algorithm, it will be useful to represent the latent values and the algorithm's decisions as matrices and vectors. For this purpose, we will number the arms as $x_1, \ldots, x_m$ and let $r \in \mathbb{R}^m$ denote the vector whose j^{th} component is $r(x_j, \mu)$. Similarly we will let $C \in \mathbb{R}^{d \times m}$ denote the matrix whose $(i,j)^{\text{th}}$ entry is $c_i(x_j, \mu)$.

2.1 High-probability events

We will use the following expression, which we call the *confidence radius*.

$$\texttt{rad}(\nu, N) = \sqrt{\frac{C_{\texttt{rad}}\,\nu}{N}} + \frac{C_{\texttt{rad}}}{N}. \tag{2}$$

Here $C_{\texttt{rad}} = \Theta(\log(dT|X|))$ is a parameter which we will fix later; we will keep it implicit in the notation. The meaning of Equation (2) and $C_{\texttt{rad}}$ is explained by the following tail inequality from [29, 9].[1]

Theorem 2.1 ([29, 9]). *Consider some distribution with values in $[0,1]$ and expectation ν. Let $\widehat{\nu}$ be the average of N independent samples from this distribution. Then*

$$\Pr[\ |\nu - \widehat{\nu}| \leq \texttt{rad}(\widehat{\nu}, N) \leq 3\,\texttt{rad}(\nu, N)\ ] \geq 1 - e^{-\Omega(C_{\texttt{rad}})}, \quad \textit{for each } C_{\texttt{rad}} > 0. \tag{3}$$

More generally, Equation (3) holds if $X_1, \ldots, X_N \in [0,1]$ are random variables, $\widehat{\nu} = \frac{1}{N}\sum_{t=1}^{N} X_t$ is the sample average, and $\nu = \frac{1}{N}\sum_{t=1}^{N} \mathbb{E}[X_t \mid X_1, \ldots, X_{t-1}]$.

If the expectation ν is a latent quantity, Equation (3) allows us to estimate ν by a high-confidence interval

$$\nu \in [\widehat{\nu} - \texttt{rad}(\widehat{\nu}, N),\ \widehat{\nu} + \texttt{rad}(\widehat{\nu}, N)], \tag{4}$$

whose endpoints are observable (known to the algorithm). For small ν, such estimate is much sharper than the one provided by Azuma-Hoeffding inequality.[2] Our algorithm uses several estimates of this form. For brevity, we say that $\widehat{\nu}$ is an N-*strong estimate* of ν if $\nu, \widehat{\nu}$ satisfy Equation (4).

It is sometimes useful to argue about *any* ν which lies in the high-confidence interval (4), not just the latent $\nu = \mathbb{E}[\widehat{\nu}]$. We use the following claim which is implicit in [29].

[1] Specifically, this follows from Lemma 4.9 in the full version of [29] (on arxiv.org), and Theorem 4.8 and Theorem 4.10 in the full version of [9] (on arxiv.org).

[2] Essentially, Azuma-Hoeffding inequality states that $|\nu - \widehat{\nu}| \leq O(\sqrt{C_{\texttt{rad}}/N})$, whereas by Theorem 2.1 for small ν it holds with high probability that $\texttt{rad}(\widehat{\nu}, N) \sim C_{\texttt{rad}}/N$.

Claim 2.2 ([29]). *For any $\nu, \widehat{\nu} \in [0,1]$, Equation (4) implies that* $\texttt{rad}(\widehat{\nu}, N) \leq 3\,\texttt{rad}(\nu, N)$.

2.2 LP-relaxation

The expected reward of the optimal policy, given foreknowledge of the distribution of outcome vectors, is typically difficult to characterize exactly. In fact, even for a time-invariant policy, it is difficult to give an exact expression for the expected reward due to the dependence of the reward on the random stopping time, τ, when the resource budget is exhausted. To approximate these quantities, we consider the fractional relaxation of BwK in which the stopping time (i.e., the total number of rounds) can be fractional, and the reward and resource consumption per unit time are deterministically equal to the corresponding expected values in the original instance of BwK.

The following LP (shown here along with its dual) constitutes our fractional relaxation of the optimal policy. We

$$\begin{array}{lll} \max & r^\intercal \xi & \\ \text{s.t.} & C\xi \preceq B\mathbf{1} & \\ & \xi \succeq 0 & \end{array} \qquad (\text{P}) \qquad\qquad \begin{array}{lll} \min & B\mathbf{1}^\intercal \eta & \\ \text{s.t.} & C^\intercal \eta \succeq r & \\ & \eta \succeq 0 & \end{array} \qquad (\text{D})$$

denote by $\texttt{OPT}_{\texttt{LP}}$ the value of the linear program (P).

Lemma 2.3. $\texttt{OPT}_{\texttt{LP}}$ *is an upper bound on the value of the optimal dynamic policy.*

Proof. One way to prove the lemma is to define ξ_j to be the expected number of times arm x_j is played by the optimal dynamic policy, and argue that ξ is primal-feasible and that $r^\intercal \xi$ is the expected reward of the optimal policy. We instead present a simple proof using the dual LP (D), since it introduces ideas that motivate the design of our primal-dual algorithm.

Let η^* denote an optimal solution to (D). By strong duality, $B\mathbf{1}^\intercal \eta^* = \texttt{OPT}_{\texttt{LP}}$. Interpret η_i^* as a unit cost for resource i. Dual feasibility implies that for each arm x_j, the expected cost of resources consumed when x_j is pulled exceeds the expected reward produced. Thus, if we let Z_t denote the sum of rewards gained in rounds $1, \ldots, t$ plus the cost of the remaining resource endowment after round t, then the stochastic process $Z_0, Z_1, \ldots, Z_T$ is a supermartingale. Note that $Z_0 = B\mathbf{1}^\intercal \eta^*$ is the LP optimum, and $Z_{\tau-1}$ equals the algorithm's total payoff, plus the cost of the remaining (non-negative) resource supply at the start of round τ. By Doob's optional stopping theorem, $Z_0 \geq \mathbb{E}[Z_{\tau-1}]$ and the lemma is proved. □

In a similar way, the expected total reward of time-invariant policy $\mathcal{D}$ is bounded above by the solution to the following linear program in which t is the only LP variable:

$$\begin{array}{lrcll} \text{Maximise} & t\, r(\mathcal{D},\mu) & & & \text{in } t \in \mathbb{R} \\ \text{subject to} & t\, c_i(\mathcal{D},\mu) & \leq & B & \text{for each resource } i \\ & t & \geq & 0. & \end{array} \qquad (5)$$

The solution to (5), which we call the *LP-value*, is

$$\texttt{LP}(\mathcal{D},\mu) = r(\mathcal{D},\mu)\ \min_i \left(\frac{B}{c_i(\mathcal{D},\mu)} \right). \qquad (6)$$

Observe that t is feasible for $\texttt{LP}(\mathcal{D},\mu)$ if and only if $\xi = t\mathcal{D}$ is feasible for (P), and therefore

$$\texttt{OPT}_{\texttt{LP}} = \sup_{\mathcal{D}} \texttt{LP}(\mathcal{D},\mu).$$

A distribution $D^* \in \operatorname{argmax}_{\mathcal{D}} \texttt{LP}(\mathcal{D},\mu)$; is called *LP-optimal* for latent structure μ. Any optimal solution ξ to (P) corresponds to an LP-optimal distribution $D^* = \xi / \|\xi\|_1$.

3 A primal-dual algorithm for BwK

This section develops an algorithm that solves the BwK problem using a very natural and intuitive idea: greedily select arms with the greatest estimated "bang per buck," i.e. reward per unit of resource consumption. One of the main

difficulties with this idea is that there is no such thing as a "unit of resource consumption": there are d different resources, and it is unclear how to trade off consumption of one resource versus another. The proof of Lemma 2.3 gives some insight into how to quantify this trade-off: an optimal dual solution η^* can be interpreted as a vector of unit costs for resources, such that for every arm the expected reward is less than or equal to the expected cost of resources consumed. Since our goal is to match the optimum value of the LP as closely as possible, we must minimize the shortfall between the expected reward of the arms we pull and their expected resource cost as measured by η^*. Thus, our algorithm will try to learn an optimal dual vector η^* in tandem with learning the latent structure μ.

Borrowing an idea from [5, 21, 35], we will use the multiplicatives weights update method to learn the optimal dual vector. This method raises the cost of a resource exponentially as it is consumed, which ensures that heavily demanded resources become costly, and thereby promotes balanced resource consumption. Meanwhile, we still have to ensure (as in any multi-armed bandit problem) that our algorithm explores the different arms frequently enough to gain adequately accurate estimates of the latent structure. We do this by estimating rewards and resource consumption as optimistically as possible, i.e. using upper confidence bound (UCB) estimates for rewards and lower confidence bound (LCB) estimates for resource consumption. Although both of these techniques — multiplicative weights and confidence bounds — have both successfully applied in previous online learing algorithms, it is far from obvious that this particular hybrid of the two methods should be effective. In particular, the use of multiplicative udpates on dual variables, rather than primal ones, distinguishes our algorithm from other bandit algorithms that use multiplicative weights (e.g. the `Exp3` algorithm [7]) and brings it closer in spirit to the literature on stochastic packing algorithms (especially [18]).

The pseudocode for the algorithm is presented as Algorithm 1, which we call `PD-BwK`. When we refer to the UCB or LCB for a latent parameter (the reward of an arm, or the amount of some resource that it utilizes), these are computed as follows. Letting $\hat{\nu}$ denote the empirical average of the observations of that random variable[3] and letting N denote the number of times the random variable has been observed, the lower confidence bound (LCB) and upper confidence bound (UCB) are the left and right endpoints, respectively, of the *confidence interval* $[0,1] \cap [\hat{\nu} - \texttt{rad}(\hat{\nu}, N),\ \hat{\nu} + \texttt{rad}(\hat{\nu}, N)]$. The UCB or LCB for a vector or matrix are defined componentwise.

In step 8, the pseudocode asserts that the set $\operatorname{argmin}_{z \in \boldsymbol{\Delta}[X]} \{\frac{y_t^\intercal L_t z}{u_t^\intercal z}\}$ contains at least one of the point-mass distributions $\{\mathbf{e}_1, \ldots, \mathbf{e}_m\}$. This is true, because if $\rho = \min_{z \in \boldsymbol{\Delta}[X]} \{(y_t^\intercal L_t z)/(u_t^\intercal z)\}$ then the linear inequality $y_t^\intercal L_t z \leq \rho u_t^\intercal \mathcal{D}$ holds at some point $z \in \boldsymbol{\Delta}[X]$, and hence it holds at some extreme point, i.e. one of the point-mass distributions.

Another feature of our algorithm that deserves mention is a variant of the Garg-Könemann width reduction technique [21]. The ratio $(y_t^\intercal L_t z_t)/(u_t^\intercal z_t)$ that we optimize in step 8 may be unboundedly large, so in the multiplicative update in step 9 we rescale this value to $y_t^\intercal L_t z_t$, which is guaranteed to be at most 1. This rescaling is mirrored in the analysis of the algorithm, when we define the dual vector $\bar{y}$ by averaging the vectors y_t using the aforementioned scale factors. (Interestingly, unlike the Garg-Könemann algorithm which applies multiplicative updates to the dual vectors and weighted averaging to the primal ones, in our algorithm the multiplicative updates and weighted averaging are *both* applied to the dual vectors.)

Algorithm 1 Algorithm `PD-BwK`

1: Set $\epsilon = \sqrt{\ln(d)/B}$.
2: In the first m rounds, pull each arm once.
3: $v_1 = \mathbf{1}$
4: **for** $t = m+1, \ldots, \tau$ *(i.e., until resource budget exhausted)* **do**
5: Compute UCB estimate for reward vector, u_t.
6: Compute LCB estimate for resource consumption matrix, L_t.
7: $y_t = v_t/(\mathbf{1}^\intercal v_t)$.
8: Pull arm x_j such that point-mass distribution $z_t = \mathbf{e}_j$ belongs to $\operatorname{argmin}_{z \in \boldsymbol{\Delta}[X]} \{\frac{y_t^\intercal L_t z}{u_t^\intercal z}\}$.
9: $v_{t+1} = \operatorname{Diag}\{(1+\epsilon)^{\mathbf{e}_i^\intercal L_t z_t}\} v_t$.

Theorem 3.1. *For any instance of the `BwK` problem, the regret of Algorithm `PD-BwK` satisfies*

$$\texttt{OPT} - \texttt{REW} \leq O\left(\sqrt{\log(dmT)}\right)\left(\sqrt{m\,\texttt{OPT}} + \texttt{OPT}\sqrt{\frac{m}{B}} + m\sqrt{\log(dmT)}\right). \tag{7}$$

[3] Note that we initialize the algorithm by pulling each arm once, so empirical averages are always well-defined.

Acknowledgements

The authors wish to thank Moshe Babaioff, Peter Frazier, Luyi Gui, Chien-Ju Ho and Jennifer Wortman Vaughan for helpful discussions related to this work. In particular, The application of dynamic procurement to crowdsourcing have been suggested to us by Chien-Ju Ho and Jennifer Wortman Vaughan.

References

[1] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. *Advances in Neural Information Processing Systems (NIPS)*, 24:2312–2320, 2011.

[2] Alekh Agarwal, Dean P. Foster, Daniel Hsu, Sham Kakade, and Alexander Rakhlin. Stochastic convex optimization with bandit feedback. *Advances in Neural Information Processing Systems (NIPS)*, 24:1035–1043, 2011.

[3] Rajeev Agrawal. The continuum-armed bandit problem. *SIAM J. Control and Optimization*, 33(6):1926–1951, 1995.

[4] Shipra Agrawal, Zizhuo Wang, and Yinyu Ye. A dynamic near-optimal algorithm for online linear programming, 2009. Technical report. Available from arXiv at `http://arxiv.org/abs/0911.2974`.

[5] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

[6] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002. Preliminary version in *15th ICML*, 1998.

[7] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, 2002. Preliminary version in *36th IEEE FOCS*, 1995.

[8] Peter Auer, Ronald Ortner, and Csaba Szepesvári. Improved Rates for the Stochastic Continuum-Armed Bandit Problem. In *20th Conf. on Learning Theory (COLT)*, pages 454–468, 2007.

[9] Moshe Babaioff, Shaddin Dughmi, Robert Kleinberg, and Aleksandrs Slivkins. Dynamic pricing with limited supply. In *13th ACM Conf. on Electronic Commerce (EC)*, 2012.

[10] Ashwinkumar Badanidiyuru, Robert Kleinberg, and Yaron Singer. Learning on a budget: posted price mechanisms for online procurement. In *13th ACM Conf. on Electronic Commerce (EC)*, pages 128–145, 2012.

[11] Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks. In *54th IEEE Symp. on Foundations of Computer Science (FOCS)*, 2013.

[12] Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks. A technical report on `arxiv.org`., May 2013.

[13] Omar Besbes and Assaf Zeevi. Dynamic pricing without knowing the demand function: Risk bounds and near-optimal algorithms. *Operations Research*, 57:1407–1420, 2009.

[14] Avrim Blum, Vijay Kumar, Atri Rudra, and Felix Wu. Online learning in online auctions. In *14th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 202–204, 2003.

[15] Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvari. Online Optimization in X-Armed Bandits. *J. of Machine Learning Research (JMLR)*, 12:1587–1627, 2011. Preliminary version in *NIPS 2008*.

[16] Varsha Dani, Thomas P. Hayes, and Sham Kakade. Stochastic Linear Optimization under Bandit Feedback. In *21th Conf. on Learning Theory (COLT)*, pages 355–366, 2008.

[17] Nikhil R. Devanur and Thomas P. Hayes. The AdWords problem: Online keyword matching with budgeted bidders under random permutations. In *Proceedings 10th ACM Conference on Electronic Commerce (EC-2009)*, pages 71–78, 2009.

[18] Nikhil R. Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A. Wilkens. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In *Proceedings 12th ACM Conference on Electronic Commerce (EC-2011)*, pages 29–38, 2011.

[19] Jon Feldman, Monika Henzinger, Nitish Korula, Vahab S. Mirrokni, and Clifford Stein. Online stochastic packing applied to display ad allocation. In *Proc. 18th European Symposium on Algorithms (ESA)*, pages 182–194, 2010.

[20] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[21] Naveen Garg and Jochen Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM J. Computing*, 37(2):630–652, 2007.

[22] Sudipto Guha and Kamesh Munagala. Multi-armed bandits with metric switching costs. In *Proc. 36th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 496–507, 2009.

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

[24] Elad Hazan and Nimrod Megiddo. Online Learning with Prior Information. In *20th Conf. on Learning Theory (COLT)*, pages 499–513, 2007.

[25] Robert Kleinberg. Nearly tight bounds for the continuum-armed bandit problem. In *18th Advances in Neural Information Processing Systems (NIPS)*, 2004.

[26] Robert Kleinberg. Lecture notes for CS 683 (week 2), Cornell University, 2007. `http://www.cs.cornell.edu/courses/cs683/2007sp/lecnotes/week2.pdf`.

[27] Robert Kleinberg and Tom Leighton. The value of knowing a demand curve: Bounds on regret for online posted-price auctions. In *44th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 594–605, 2003.

[28] Robert Kleinberg and Aleksandrs Slivkins. Sharp Dichotomies for Regret Minimization in Metric Spaces. In *21st ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 2010.

[29] Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Multi-Armed Bandits in Metric Spaces. In *40th ACM Symp. on Theory of Computing (STOC)*, pages 681–690, 2008.

[30] T. L. Lai and Herbert Robbins. Asymptotically efficient adaptive allocations rules. *Adv. in Appl. Math.*, 6:4–22, 1985.

[31] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–260, 1994.

[32] Tyler Lu, Dávid Pál, and Martin Pál. Showing Relevant Ads via Lipschitz Context Multi-Armed Bandits. In *14thIntl. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2010.

[33] Marco Molinaro and R. Ravi. Geometry of online packing linear programs. In *Proc. 39th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 701–713, 2012.

[34] Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of optimal queuing network control. *Math. Oper. Res.*, 24(2):293–305, 1999.

[35] Serge A. Plotkin, David B. Shmoys, and Eva Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20:257–301, 1995.

[36] Long Tran-Thanh. *Budget-Limited Multi-Armed Bandits*. PhD thesis, University of Southampton, 2012.

[37] Long Tran-Thanh, Archie Chapman, Enrique Munoz de Cote, Alex Rogers, and Nicholas R. Jennings. ϵ-first policies for budget-limited multi-armed bandits. In *Proc. Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)*, pages 1211–1216, 2010.

[38] Long Tran-Thanh, Archie Chapman, Alex Rogers, and Nicholas R. Jennings. Knapsack based optimal policies for budget-limited multi-armed bandits. In *Proc. Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI-12)*, pages 1134–1140, 2012.

[39] Peter Whittle. Multi-armed bandits and the Gittins index. *J. Royal Statistical Society, Series B*, 42(2):143–149, 1980.

Appendix A: Optimal dynamic policy beats the best fixed arm

Let us provide some examples of `BwK` problem instances in which the best dynamic policy (in fact, the best fixed distribution over arms) beats the best fixed arm.

There are d arms; pulling arm i deterministically produces a reward of 1, consumes one unit of resource i, and does not consume any other resources. We are given an initial endowment of B units of each resource. Any policy that plays a fixed arm i in each period is limited to a total reward of B before running out of its budget of resource i. A policy that always plays a uniformly random arm achieves an expected reward of nearly $d \cdot B$ before its resource budget runs out.

Less contrived examples of the same phenomenon arises in dynamic procurement. Consider the basic setting of "dynamic procurement": in each round a potential seller arrives, and the buyer offers to buy one item at a price; there are T sellers and the buyer is constrained to spend atmost budget B. The buyer has no value for left-over budget and each sellers value for the item is drawn i.i.d from an unknown distribution. One can easily construct distributions for which offering a mixture of two prices is strictly superior to offering any fixed price. In fact this situation arises whenever the "sales curve" (the mapping from prices to probability of selling) is non-concave and its value at the quantile B/T lies below its concave hull. We provide a specific example below.

The problem instance is defined as follows. Fix any constant $\delta > 0$, and let $\epsilon = B^{1/2+\delta}$. Each seller has the following two-point demand distribution: the seller's value for item is $v = 1$ with probability $\frac{B}{T}$, and $v = 0$ with the remaining

probability. Let $\texttt{REW}(\mathcal{D})$ be the total expected number of sales from using a fixed distribution $\mathcal{D}$ over prices in each round; let $\texttt{REW}(p)$ be the same quantity when $\mathcal{D}$ deterministically picks a given price p.

- Clearly, if one offers a fixed price in all rounds, it only makes sense to offer prices $p = 0$ and $p = 1$. It is easy to see that $\texttt{REW}(0) <= T \cdot \mathbb{E}[\text{Probability of selling at price } 0] = B.$ and $\texttt{REW}(1) = B$.
- Now consider a distribution $\mathcal{D}$ which picks price 0 with probability $1 - \frac{B-\epsilon}{T}$, and picks price 1 with the remaining probability. It is easy to show that $\texttt{REW}(D) \geq (2 - o(1))\, B$.

So, $\texttt{REW}(\mathcal{D})$ is essentially twice as large compared to the total expected sales of the best fixed arm.

Appendix B: Analysis of the Hedge Algorithm

In this section we analyze the Hedge algorithm, also known as the multiplicative weights algorithm. It is an online algorithm for maintaining a d-dimensional probability vector y while observing a sequence of d-dimensional payoff vectors $\pi_1, \ldots, \pi_\tau$. The algorithm is initialized with a parameter $\epsilon \in (0, 1)$.

Algorithm 2 The algorithm Hedge(ϵ)

1: $v_1 = \mathbf{1}$
2: **for** $t = 1, 2, \ldots, \tau$ **do**
3: $\quad y_t = v_t/(\mathbf{1}^\intercal v_t)$.
4: $\quad v_{t+1} = \mathrm{Diag}\{(1+\epsilon)^{\pi_{ti}}\} v_t$.

The performance guarantee of the algorithm is expressed by the following proposition.

Proposition B.1. *For any $0 < \epsilon < 1$ and any sequence of payoff vectors $\pi_1, \ldots, \pi_\tau \in [0,1]^d$, we have*

$$\forall y \in \mathbf{\Delta}[d] \quad \sum_{t=1}^{\tau} y_t^\intercal \pi_t \geq (1-\epsilon) \sum_{t=1}^{\tau} y^\intercal \pi_t - \frac{\ln d}{\epsilon}.$$

Proof. The analysis uses the potential function $\Phi_t = \mathbf{1}^\intercal v_t$. We have

$$\begin{aligned}
\Phi_{t+1} &= \mathbf{1}^\intercal \mathrm{Diag}\{(1+\epsilon)^{\pi_{ti}}\} v_t \\
&= \sum_{i=1}^{d} (1+\epsilon)^{\pi_{ti}} v_t \\
&\leq \sum_{i=1}^{d} (1 + \epsilon \pi_{ti}) v_t \\
&= \Phi_t \left(1 + \epsilon y_t^\intercal \pi_t\right) \\
\ln(\Phi_{t+1}) &\leq \ln(\Phi_t) + \ln(1 + \epsilon y_t^\intercal \pi_t) \leq \ln(\Phi_t) + \epsilon y_t^\intercal \pi_t.
\end{aligned}$$

On the third line, we have used the inequality $(1+\epsilon)^x \leq 1 + \epsilon x$ which is valid for $0 \leq x \leq 1$. Now, summing over $t = 1, \ldots, \tau$ we obtain

$$\sum_{t=1}^{\tau} y_t^\intercal \pi_t \geq \tfrac{1}{\epsilon} \left(\ln \Phi_{\tau+1} - \ln \Phi_1\right) = \tfrac{1}{\epsilon} \ln \Phi_{\tau+1} - \frac{\ln d}{\epsilon}.$$

The maximum of $y^\intercal \left(\sum_{t=1}^{\tau} \pi_t\right)$ over $y \in \mathbf{\Delta}[d]$ must be attained at one of the extreme points of $\mathbf{\Delta}[d]$, which are simply the standard basis vectors of $\mathbb{R}^d$. Say that the maximum is attained at $\mathbf{e}_i$. Then we have

$$\begin{aligned}
\Phi_{\tau+1} = \mathbf{1}_\intercal v_{\tau+1} \geq v_{\tau+1,i} &= (1+\epsilon)^{\pi_{1i}+\cdots+\pi_{\tau i}} \\
\ln \Phi_{\tau+1} &\geq \ln(1+\epsilon) \sum_{t=1}^{\tau} \pi_{ti} \\
\sum_{t=1}^{\tau} y_t^\intercal \pi_t &\geq \frac{\ln(1+\epsilon)}{\epsilon} \sum_{t=1}^{\tau} \pi_{ti} - \frac{\ln d}{\epsilon} \\
&\geq (1-\epsilon) \sum_{t=1}^{\tau} y^\intercal \pi_t - \frac{\ln d}{\epsilon}.
\end{aligned}$$

The last line follows from two observations. First, our choice of i ensures that $\sum_{t=1}^{\tau} \pi_{ti} \geq \sum_{t=1}^{\tau} y^\intercal \pi_t$ for every $y \in \mathbf{\Delta}[d]$. Second, the inequality $\ln(1+\epsilon) > \epsilon - \epsilon^2$ holds for every $\epsilon > 0$. In fact,

$$\begin{aligned}
-\ln(1+\epsilon) = \ln\left(\frac{1}{1+\epsilon}\right) = \ln\left(1 - \frac{\epsilon}{1+\epsilon}\right) &< -\frac{\epsilon}{1+\epsilon} \\
\ln(1+\epsilon) > \frac{\epsilon}{1+\epsilon} &> \frac{\epsilon(1-\epsilon^2)}{1+\epsilon} = \epsilon - \epsilon^2.
\end{aligned}$$

□

Appendix C: LP-optimal distributions over arms

We flesh out a claim about LP-optimal distributions, which we use for discretization (Section E.1).

Claim C.1. *For any latent structure μ, there exists a distribution $\mathcal{D}$ over arms which is LP-optimal for μ and moreover satisfies the following three properties:*

(a) $c_i(\mathcal{D}, \mu) \leq B/T$ for each resource i.
(b) $\mathcal{D}$ has a support of size at most $d+1$.
(c) If $\mathcal{D}$ has a support of size at least 2 then for some resource i we have $c_i(\mathcal{D}, \mu) = B/T$.

*(Such distribution $\mathcal{D}$ will be called **LP-perfect** for μ.)*

Proof. Fix the latent structure μ. It is a well-known fact that for any linear program there exists an optimal solution whose support has size that is exactly equal to the number of constraints that are tight for this solution. Take any such optimal solution ξ for linear program (P), and take the corresponding LP-optimal distribution $\mathcal{D} = \xi/\|\xi\|_1$. Since there are $d+1$ constraints in (P), distribution $\mathcal{D}$ has support of size at most $d+1$. If it satisfies property (a), we are done. Note that such $\mathcal{D}$ also satisfies property (c).

Suppose property (a) does not hold for $\mathcal{D}$. Then there exists a resource i such that $c_i(\mathcal{D}, \mu) > B/T$. It follows that $\sum_i \xi_i < T$. Therefore, at most d constraints in (P) are tight for ξ, which implies that the support of $\mathcal{D}$ has size at most d.

Now, let us modify $\mathcal{D}$ to obtain another LP-optimal distribution $\mathcal{D}'$ which satisfies both (a) and (b-c). Namely, let $\mathcal{D}'(x) = \alpha\, \mathcal{D}(x)$ for each non-null arm x, for some $\alpha \in (0,1)$ that is sufficiently low to ensure property (a) (with equality for some resource i), and place the remaining probability in $\mathcal{D}'$ on the null arm. Then $\texttt{LP}(\mathcal{D}', \mu) = \texttt{LP}(\mathcal{D}, \mu)$, so $\mathcal{D}'$ is LP-optimal; $\mathcal{D}'$ satisfies properties (a,c) by design, and it satisfies property (b) because it adds at most one to the support of $\mathcal{D}$. □

Appendix D: Analysis of PD−BwK (Proof of Theorem 3.1)

In this section we present the analysis of the PD−BwK algorithm. We begin by recalling the Hedge algorithm from online learning theory, and its performance guarantee. Next, we present a simplified analysis of PD−BwK in a toy model in which the outcome vectors are deterministic. (This toy model of the BwK problem is uninteresting as a

learning problem since the latent structure does not need to be learned, and the problem reduces to solving a linear program. We analyze PD-BwK first in this context because the analysis is simple, yet its main ideas carry over to the general case which is more complicated.)

D.1 The Hedge algorithm

In this section we present the Hedge algorithm, also known as the multiplicative weights algorithm. It is an online algorithm for maintaining a d-dimensional probability vector y while observing a sequence of d-dimensional payoff vectors $\pi_1, \ldots, \pi_\tau$. The algorithm is initialized with a parameter $\epsilon \in (0, 1)$.

Algorithm 3 The algorithm Hedge(ϵ)

1: $v_1 = \mathbf{1}$
2: **for** $t = 1, 2, \ldots, \tau$ **do**
3: $\quad y_t = v_t/(\mathbf{1}^\intercal v_t)$.
4: $\quad v_{t+1} = \mathrm{Diag}\{(1+\epsilon)^{\pi_{ti}}\}v_t$.

The Hedge algorithm is due to Freund and Schapire [20]. The version presented above, along with the following performance guarantee, are adapted from [26].

Proposition D.1. *For any $0 < \epsilon < 1$ and any sequence of payoff vectors $\pi_1, \ldots, \pi_\tau \in [0, 1]^d$, we have*

$$\forall y \in \mathbf{\Delta}[d] \quad \sum_{t=1}^{\tau} y_t^\intercal \pi_t \geq (1-\epsilon) \sum_{t=1}^{\tau} y^\intercal \pi_t - \frac{\ln d}{\epsilon}.$$

A self-contained proof of the proposition, for the reader's convenience, appears in Appendix B.

D.2 Warm-up: The deterministic case

To present the application of Hedge to BwK in its purest form, we first present an algorithm for the case in which the rewards of the various arms are deterministically equal to the components of a vector $r \in \mathbb{R}^m$, and the resource consumption vectors are deterministically equal to the columns of a matrix $C \in \mathbb{R}^{d \times m}$.

Algorithm 4 Algorithm PD-BwK, adapted for deterministic outcomes

1: In the first m rounds, pull each arm once.
2: Let r, C denote the reward vector and resource consumption matrix revealed in Step 1.
3: $v_1 = \mathbf{1}$
4: $t = 1$
5: **for** $t = m+1, \ldots, \tau$ *(i.e., until resource budget exhausted)* **do**
6: $\quad y_t = v_t/(\mathbf{1}^\intercal v_t)$.
7: $\quad$ Pull arm x_j such that point-mass distribution $z_t = \mathbf{e}_j$ belongs to $\mathrm{argmin}_{z \in \mathbf{\Delta}[X]}\{\frac{y_t^\intercal C z}{r^\intercal z}\}$.
8: $\quad v_{t+1} = \mathrm{Diag}\{(1+\epsilon)^{\mathbf{e}_i^\intercal C z_t}\}v_t$.
9: $\quad t = t+1$.

This algorithm is an instance of the multiplicative-weights update method for solving packing linear programs. Interpreting it through the lens of online learning, as in the survey [5], it is updating a vector y using the Hedge algorithm. The payoff vector in round t is given by Cx_t. Note that these payoffs belong to $[0, 1]$, by our assumption that C has $[0, 1]$ entries.

Now let ξ^* denote an optimal solution of the primal linear program (P) from Section 2.2, and let $\mathtt{OPT}_{\mathtt{LP}} = r^\intercal \xi^*$ denote the optimal value of that LP. Let $\mathtt{REW} = \sum_{t=m+1}^{\tau-1} r^\intercal z_t$ denote the total payoff obtained by the algorithm after its start-up phase (the first m rounds). By the stopping condition for the algorithm, we know there is a vector $y \in \mathbf{\Delta}[\mathscr{R}]$ such that $y^\intercal C\left(\sum_{t=1}^{\tau} z_t\right) \geq B$ so fix one such y. Note that the algorithm's start-up phase consumes at most m units of any resource, and the final round τ consumes at most one unit more, so

$$y^\intercal C \left(\sum_{m < t < \tau} z_t \right) \geq B - m - 1. \tag{8}$$

Finally let

$$\bar{y} = \frac{1}{\texttt{REW}} \sum_{m<t<\tau} (r^\intercal z_t) y_t.$$

Now we have

$$\begin{aligned} B \geq \bar{y}^\intercal C \xi^* &= \frac{1}{\texttt{REW}} \sum_{m<t<\tau} (r^\intercal z_t)(y_t^\intercal C \xi^*) \\ &\geq \frac{1}{\texttt{REW}} \sum_{m<t<\tau} (r^\intercal \xi^*)(y_t^\intercal C z_t) \\ &\geq \frac{\texttt{OPT}_{\texttt{LP}}}{\texttt{REW}} \left[(1-\epsilon) \sum_{m<t<\tau} y^\intercal C z_t - \frac{\ln d}{\epsilon} \right] \\ &\geq \frac{\texttt{OPT}_{\texttt{LP}}}{\texttt{REW}} \left[B - \epsilon B - m - 1 - \frac{\ln d}{\epsilon} \right]. \end{aligned}$$

The first inequality is by primal fesibility of ξ^*, the second is by the definition of z_t, the third is the performance guarantee of Hedge, the fourth is by (8). Putting these inequalities together we have derived

$$\frac{\texttt{REW}}{\texttt{OPT}_{\texttt{LP}}} \geq 1 - \epsilon - \frac{m+1}{B} - \frac{\ln d}{\epsilon B}.$$

Setting $\epsilon = \sqrt{\frac{\ln d}{B}}$ we find that the algorithm's regret is bounded above by $\texttt{OPT}_{\texttt{LP}} \cdot O\left(\sqrt{\frac{\ln d}{B}} + \frac{m}{B}\right)$.

D.3 Analysis modulo error terms

We now commence the analysis of Algorithm PD-BwK. In this section we show how to reduce the problem of bounding the algorithm's regret to a problem of estimating two error terms that reflect the difference between the algorithm's confidence-bound estimates of its own reward and resource consumption with the empirical values of these random variables. The analysis of those error terms will be the subject of Section D.4.

By Theorem 2.1 and our choice of $C_{\texttt{rad}}$, it holds with probability at least $1 - T^{-1}$ that the confidence interval for every latent parameter, in every round of execution, contains the true value of that latent parameter. We call this high-probability event a *clean execution* of PD-BwK. Our regret guarantee will hold deterministically assuming that a clean execution takes place. The regret can be at most T when a clean execution does not take place, and since this event has probability at most T^{-1} it contributes only $O(1)$ to the regret. We will henceforth assume a clean execution of PD-BwK.

Claim D.2. *In a clean execution of Algorithm PD-BwK, the algorithm's total reward satisfies the bound*

$$\texttt{REW} \geq \texttt{OPT}_{\texttt{LP}} - \left[2\texttt{OPT}_{\texttt{LP}} \left(\sqrt{\frac{\ln d}{B}} + \frac{m+1}{B} \right) + m + 1 \right] - \frac{\texttt{OPT}_{\texttt{LP}}}{B} \left\| \sum_{m<t<\tau} E_t z_t \right\|_\infty - \left| \sum_{m<t<\tau} \delta_t^\intercal z_t \right|, \tag{9}$$

where E_t and δ_t are defined by

$$E_t = C_t - L_t, \qquad \delta_t = u_t - r_t. \tag{10}$$

Proof. The claim is proven by mimicking the analysis of Algorithm 4 in the preceding section, incorporating error terms that reflect the differences between observable values and latent ones. As before, let ξ^* denote an optimal solution of the primal linear program (P), and let $\texttt{OPT}_{\texttt{LP}} = r^\intercal \xi^*$ denote the optimal value of that LP. Let $\texttt{REW}_{\texttt{UCB}} = \sum_{m<t<\tau} u_t^\intercal z_t$ denote the total payoff the algorithm would have obtained, after its start-up phase, if the actual payoff at time t were replaced with the upper confidence bound. By the stopping condition for the algorithm, we know there is a vector $y \in \boldsymbol{\Delta}[\mathscr{R}]$ such that $y^\intercal \left(\sum_{t=1}^{\tau} C_t z_t\right) \geq B$, so fix one such y. As before,

$$y^\intercal \left(\sum_{m<t<\tau} C_t z_t \right) \geq B - m - 1. \tag{11}$$

Finally let

$$\bar{y} = \frac{1}{\texttt{REW}_{\texttt{UCB}}} \sum_{m<t<\tau} (u_t^\intercal z_t) y_t.$$

Assuming a clean execution, we have

$$
\begin{aligned}
B &\geq \bar{y}^{\intercal} C \xi^* && \textit{(ξ^* is primal feasible)} \\
&= \frac{1}{\texttt{REW}_{\texttt{UCB}}} \sum_{m<t<\tau} (u_t^{\intercal} z_t)(y_t^{\intercal} C \xi^*) \\
&\geq \frac{1}{\texttt{REW}_{\texttt{UCB}}} \sum_{m<t<\tau} (u_t^{\intercal} z_t)(y_t^{\intercal} L_t \xi^*) && \textit{(clean execution)} \\
&\geq \frac{1}{\texttt{REW}_{\texttt{UCB}}} \sum_{m<t<\tau} (u_t^{\intercal} \xi^*)(y_t^{\intercal} L_t z_t) && \textit{(definition of z_t)} \\
&\geq \frac{1}{\texttt{REW}_{\texttt{UCB}}} \sum_{m<t<\tau} (r^{\intercal} \xi^*)(y_t^{\intercal} L_t z_t) && \textit{(clean execution)} \\
&\geq \frac{\texttt{OPT}_{\texttt{LP}}}{\texttt{REW}_{\texttt{UCB}}} \left[(1-\epsilon) y^{\intercal} \left(\sum_{m<t<\tau} L_t z_t \right) - \frac{\ln d}{\epsilon} \right] && \textit{(Hedge guarantee)} \\
&= \frac{\texttt{OPT}_{\texttt{LP}}}{\texttt{REW}_{\texttt{UCB}}} \left[(1-\epsilon) y^{\intercal} \left(\sum_{m<t<\tau} C_t z_t \right) - (1-\epsilon) y^{\intercal} \left(\sum_{m<t<\tau} E_t z_t \right) - \frac{\ln d}{\epsilon} \right] \\
&\geq \frac{\texttt{OPT}_{\texttt{LP}}}{\texttt{REW}_{\texttt{UCB}}} \left[(1-\epsilon) B - m - 1 - (1-\epsilon) y^{\intercal} \left(\sum_{m<t<\tau} E_t z_t \right) - \frac{\ln d}{\epsilon} \right] && \textit{(definition of y; see eq. (11))}
\end{aligned}
$$

$$
\texttt{REW}_{\texttt{UCB}} \geq \texttt{OPT}_{\texttt{LP}} \left[1 - \epsilon - \frac{m+1}{B} - \frac{1}{B} \left\| \sum_{m<t<\tau} E_t z_t \right\|_{\infty} - \frac{\ln d}{\epsilon B} \right]. \tag{12}
$$

The algorithm's actual payoff, $\texttt{REW} = \sum_{t=1}^{\tau} r_t^{\intercal} z_t$, satisfies the inequality

$$
\texttt{REW} \geq \texttt{REW}_{\texttt{UCB}} - \sum_{m<t<\tau} (u_t - r_t)^{\intercal} z_t = \texttt{REW}_{\texttt{UCB}} - \sum_{m<t<\tau} \delta_\tau^{\intercal} z_t.
$$

Combining this inequality with (12), we obtain the bound (9), as claimed. □

D.4 Error analysis

In this section we complete the proof of Theorem 3.1 by deriving upper bounds on the terms $\left\|\sum_{m<t<\tau} E_t z_t\right\|_\infty$ and $\left|\sum_{m<t<\tau} \delta_t z_t\right|$ appearing on the right side of (9). Both bounds are special cases of a more general statement, presented as Lemma D.4 below. Before stating the lemma, we need to establish a simple fact about confidence radii.

Lemma D.3. *For any two vectors $a, N \in \mathbb{R}_+^m$, we have*

$$
\sum_{j=1}^{m} \texttt{rad}(a_j, N_j)\, N_j \leq \sqrt{C_{\texttt{rad}}\, m (a^{\intercal} N)} + C_{\texttt{rad}}\, m. \tag{13}
$$

Proof. The definition of $\texttt{rad}(\cdot,\cdot)$ implies that $\texttt{rad}(a_j, N_j)\, N_j \leq \sqrt{C_{\texttt{rad}}\, a_j N_j} + C_{\texttt{rad}}$. Summing these inequalities and applying Cauchy-Schwarz,

$$
\sum_{j=1}^{m} \texttt{rad}(a_j, N_j)\, N_j \leq \sum_{j=1}^{m} \sqrt{C_{\texttt{rad}}\, a_j N_j} + C_{\texttt{rad}}\, m \leq \sqrt{m} \cdot \sqrt{\sum_{j \in X} C_{\texttt{rad}}\, a_j N_j} + C_{\texttt{rad}}\, m,
$$

and the lemma follows by rewriting the expression on the right side. □

The next lemma requires some notation and definitions. Suppose that $a_1, \ldots, a_\tau$ is a sequence of vectors in $[0,1]^m$ and that $z_1, \ldots, z_\tau$ is a sequence of vectors in $\{\mathbf{e}_1, \ldots, \mathbf{e}_m\}$. Assume that the latter sequence begins with a permutation of the elements of $\{\mathbf{e}_1, \ldots, \mathbf{e}_m\}$ so that, for every $s \geq m$, the diagonal matrix $\sum_{t=1}^{s} z_t z_t^{\intercal}$ is invertible. The vector

$$
\overline{a}_s = \left(\sum_{t=1}^{s} z_t z_t^{\intercal} \right)^{-1} \left(\sum_{t=1}^{s} z_t z_t^{\intercal} a_t \right)
$$

can be interpreted as the vector of empirical averages: the entry $\overline{a}_{s,j}$ is equal to the average of $a_{t,j}$ over all times $t \le s$ such that $z_t = \mathbf{e}_j$. Let $\mathbf{N}_s = \sum_{t=1}^{s} z_t$ denote the vector whose entry $\mathbf{N}_{s,j}$ indicates the number of times $\mathbf{e}_j$ occurs in the sequence $z_1, \ldots, z_s$. A useful identity is

$$\overline{a}_s{}^\intercal \mathbf{N}_t = \sum_{s=1}^{t} a_s^\intercal z_s.$$

Lemma D.4. *Suppose we are given sequences of vectors $a_1, \ldots, a_\tau$ and $z_1, \ldots, z_\tau$ as above. Suppose we are additionally a sequence of vectors $b_1, \ldots, b_\tau$ in $[0,1]^m$ and another vector a_0 such that for $m < t < \tau$ and $j \in X$,*

$$|b_{t,j} - a_{0,j}| \le 2\,\mathtt{rad}(\overline{a}_{t,j}, N_{t,j}) \le 6\,\mathtt{rad}(a_{0,j}, N_{t,j}).$$

Let $s = \tau - 1$ and suppose that for all $j \in X$, $|\overline{a}_{s,j} - a_{0,j}| \le \mathtt{rad}(\overline{a}_{s,j}, N_{s,j})$. Then

$$\left| \sum_{m<t<\tau} (b_t - a_t)^\intercal z_t \right| \le O\left(\sqrt{C_{\mathtt{rad}}\, mA} + C_{\mathtt{rad}}\, m \right), \tag{14}$$

where $A = (\overline{a}_{\tau-1})^\intercal \mathbf{N}_{\tau-1} = \sum_{t=1}^{\tau-1} a_t^\intercal z_t$.

Proof. The proof decomposes the left side of (14) as a sum of three terms,

$$\sum_{m<t<\tau} (b_t - a_t)^\intercal z_t = \sum_{t=1}^{m} (a_t - b_t)^\intercal z_t + \sum_{t=1}^{\tau-1} (b_t - a_0)^\intercal z_t + \sum_{t=1}^{\tau-1} (a_0 - a_t)^\intercal z_t, \tag{15}$$

then bounds the three terms separately. The first sum is clearly bounded above by m. We next work on bounding the third sum. Let $s = \tau - 1$.

$$\begin{aligned}
\sum_{t=1}^{s} (a_0 - a_t)^\intercal z_t &= a_0^\intercal \mathbf{N}_t - \sum_{t=1}^{s} a_t^\intercal z_t = (a_0 - \overline{a}_s)^\intercal \mathbf{N}_s \\
\left| \sum_{t=1}^{s} (a_0 - a_t)^\intercal z_t \right| &= |(a_0 - \overline{a}_s)^\intercal \mathbf{N}_s| \le \sum_{j \in X} \mathtt{rad}(\overline{a}_{s,j}, N_{s,j}) N_{s,j} \\
&\le \sqrt{C_{\mathtt{rad}}\, mA} + C_{\mathtt{rad}}\, m,
\end{aligned}$$

where the last line follows from Lemma D.3.

Finally we bound the middle sum in (15).

$$\begin{aligned}
\left| \sum_{t=1}^{s} (b_t - a_0)^\intercal z_t \right| &\le 6 \sum_{t=1}^{s} \sum_{j \in X} \mathtt{rad}(a_{0,j}, N_{t,j}) z_{t,j} \\
&= 6 \sum_{j \in X} \sum_{\ell=1}^{N_{s,j}} \mathtt{rad}(a_{0,j}, \ell) \\
&= O\left(\sum_{j \in X} \mathtt{rad}(a_{0,j}, N_{s,j}) N_{s,j} \right) \\
&\le O\left(\sqrt{C_{\mathtt{rad}}\, n a_0^\intercal \mathbf{N}_s} + C_{\mathtt{rad}}\, m \right).
\end{aligned}$$

We would like to replace the expression $a_0^\intercal \mathbf{N}_s$ on the last line with the expression $\overline{a}_s{}^\intercal \mathbf{N}_s = A$. To do so, recall from earlier in this proof that $|(a_0 - \overline{a}_s)^\intercal \mathbf{N}_s| \le \sqrt{C_{\mathtt{rad}}\, mA} + C_{\mathtt{rad}}\, m$, and now apply the following calculation:

$$\begin{aligned}
a_0^\intercal \mathbf{N}_s \le \overline{a}_s{}^\intercal \mathbf{N}_s + \sqrt{C_{\mathtt{rad}}\, mA} + C_{\mathtt{rad}}\, m = A + \sqrt{C_{\mathtt{rad}}\, mA} + C_{\mathtt{rad}}\, m \le \left(\sqrt{A} + \sqrt{C_{\mathtt{rad}}\, m} \right)^2 \\
\sqrt{C_{\mathtt{rad}}\, m a_0^\intercal \mathbf{N}_s} \le \sqrt{C_{\mathtt{rad}}\, m} \left(\sqrt{A} + \sqrt{C_{\mathtt{rad}}\, m} \right) = \sqrt{C_{\mathtt{rad}}\, mA} + C_{\mathtt{rad}}\, m.
\end{aligned}$$

Summing up the upper bounds for the three terms on the right side of (15), we obtain (14). □

Corollary D.5. *In a clean execution of* PD-BwK*,*

$$\left|\sum_{m<t<\tau} \delta_t z_t\right| \leq O\left(\sqrt{C_{\texttt{rad}}\, m\texttt{REW}} + C_{\texttt{rad}}\, m\right)$$

and

$$\left\|\sum_{m<t<\tau} E_t z_t\right\|_\infty \leq O\left(\sqrt{C_{\texttt{rad}}\, mB} + C_{\texttt{rad}}\, m\right).$$

Proof. The first inequality is obtained by applying Lemma D.4 with vector sequences $a_t = r_t$ and $b_t = u_t$. The second one is obtained by applying the same lemma with vector sequences $a_t = \mathbf{e}_i^\intercal C_t$ and $b_t = \mathbf{e}_i L_t$, for each resource i. □

Proof of Theorem 3.1: The theorem asserts that

$$\texttt{REW} \geq \texttt{OPT} - O\left(\sqrt{m\log(dmT)}\left(\frac{\texttt{OPT}}{\sqrt{B}} + \sqrt{\texttt{OPT}} + \sqrt{m\log(dmT)}\right)\right).$$

Denote the right-hand side by $f(\texttt{OPT})$. The function $\max\{f(x), 0\}$ is a non-decreasing function of x, so to establish the theorem we will prove that $\texttt{REW} \geq f(\texttt{OPT}_{\texttt{LP}}) \geq f(\texttt{OPT})$, where the latter inequality is an immediate consequence of the fact that $\texttt{OPT}_{\texttt{LP}} \geq \texttt{OPT}$ (Lemma 2.3).

To prove $\texttt{REW} \geq f(\texttt{OPT}_{\texttt{LP}})$, we begin by recalling inequality (9) and observing that

$$2\texttt{OPT}_{\texttt{LP}}\left(\sqrt{\frac{\ln d}{B}} + \frac{m+1}{B}\right) = O\left(\sqrt{m\log(dmT)}\,\frac{\texttt{OPT}_{\texttt{LP}}}{\sqrt{B}}\right)$$

by our assumption that $m \leq B$. The term $m+1$ on the right side of Equation (9) is bounded above by $m\log(dmT)$. Finally, using Corollary D.5 we see that the sum of the final two terms on the right side of (9) is bounded by $O\left(\sqrt{C_{\texttt{rad}}\, m}\left(\frac{\texttt{OPT}_{\texttt{LP}}}{\sqrt{B}} + \sqrt{\texttt{OPT}_{\texttt{LP}}} + \sqrt{C_{\texttt{rad}}\, m}\right)\right)$. The theorem follows by plugging in $C_{\texttt{rad}} = \Theta(\log(dmT))$. □

Appendix E: BwK with discretization

In this section we flesh out our approach to BwK with uniform discretization, as discussed in the Introduction. The technical contribution here is a definition of the appropriate structure, and proof that discretization does not do much damage.

Definition E.1. We say that arm x *ϵ-covers* arm y if the following two properties are satisfied for each resource i and every latent structure μ in a given BwK domain:

(i) $r(x,\mu)/c_i(x,\mu) \geq r(y,\mu)/c_i(y,\mu) - \epsilon$.

(ii) $c_i(x,\mu) \geq c_i(y,\mu)$.

A subset $S \subset X$ of arms is called an *ϵ-discretization* of X if each arm in X is ϵ-covered by some arm in S.

Note that we consider the difference in the ratio of expected reward to expected consumption, whereas for MAB in metric spaces it suffices to consider the difference in expected rewards.

Once we construct an ϵ-discretization S of X, we can run a BwK algorithm on S and obtain expected total reward competitive with $\texttt{OPT}_{\texttt{LP}}(S)$, the value of $\texttt{OPT}_{\texttt{LP}}$ if the action space is restricted to S. This makes sense if $\texttt{OPT}_{\texttt{LP}}(S)$ is not much worse than $\texttt{OPT}_{\texttt{LP}}(X)$.

Theorem E.2 (BwK: discretization)**.** *Fix a BwK domain with action space X. Consider an algorithm $\mathcal{A}$ which achieves expected total reward $\texttt{REW} \geq \texttt{OPT}_{\texttt{LP}}(S) - R(S)$ if applied to the restricted action space $S \subset X$, for any given $S \subset X$. If S is an ϵ-discretization of X, for some $\epsilon \geq 0$, then $\texttt{REW} \geq \texttt{OPT}_{\texttt{LP}} - \epsilon\, d\, B - R(S)$.*

The main technical result here is that $\texttt{OPT}_{\texttt{LP}}(S)$ is not much worse than $\texttt{OPT}_{\texttt{LP}}(X)$. Such result is typically trivial in MAB settings where the benchmark is the best fixed arm, but requires some work in our setting because we need to consider distributions over arms.

Lemma E.3. *Consider an instance of* BwK *with action space X. Let $S \subset X$ be an ϵ-discretization of X, for some $\epsilon \geq 0$. Then $\texttt{OPT}_{\texttt{LP}}(S) \geq \texttt{OPT}_{\texttt{LP}}(X) - \epsilon\, d\, B$*

Proof. Let $\mathcal{D}$ be the distribution over arms in X which maximizes $\texttt{LP}(\mathcal{D}, \mu)$. We use $\mathcal{D}$ to construct a distribution $\mathcal{D}_S$ over S which is nearly as good.

Let μ be the (actual) latent structure. For brevity, we will suppress μ from the notation: $c_i(x) = c_i(x, \mu)$ and $c_i(\mathcal{D}) = c_i(\mathcal{D}, \mu)$ for arms x, distributions $\mathcal{D}$ and resources i. Similarly, we will write $r(x) = r(x, \mu)$ and $r(\mathcal{D}) = r(\mathcal{D}, \mu)$.

We define $\mathcal{D}_S$ as follows. Since S is an ϵ-discretization of X, there exists a family of subsets $(\texttt{cov}(x) \subset X : x \in X)$ so that each arm x ϵ-covers all arms in $\texttt{cov}(x)$, the subsets are disjoint, and their union is X. Fix one such family, and define

$$\mathcal{D}_S(x) = \sum_{y \in \texttt{cov}(x)} \mathcal{D}(y) \min_i \frac{c_i(y)}{c_i(x)}, \quad x \in S.$$

To argue that $\texttt{LP}(\mathcal{D}_S, \mu)$ is large, we upper-bound the resource consumption $c_i(\mathcal{D}_S)$, for each resource i, and lower-bound the reward $r(\mathcal{D}_S)$.

$$\begin{aligned}
c_i(\mathcal{D}_S) &= \textstyle\sum_{x \in S} c_i(x)\, \mathcal{D}_S(x) \\
&\leq \sum_{x \in S} c_i(x) \sum_{y \in \texttt{cov}(x)} \mathcal{D}(y) \frac{c_i(y)}{c_i(x)} = \sum_{x \in S} \sum_{y \in \texttt{cov}(x)} \mathcal{D}(y)\, c_i(y) = \sum_{y \in X} \mathcal{D}(y) c_i(y) \\
&= c_i(\mathcal{D}) \qquad (16)
\end{aligned}$$

(Note that the above argument did not use the properties (i-ii) in Definition E.1.)

$$\begin{aligned}
r(\mathcal{D}_S) &= \textstyle\sum_{x \in S} r(x)\, \mathcal{D}_S(x) \\
&= \sum_{x \in S} r(x) \sum_{y \in \texttt{cov}(x)} \mathcal{D}(y) \min_i \frac{c_i(y)}{c_i(x)} \\
&= \sum_{x \in S} \sum_{y \in \texttt{cov}(x)} \mathcal{D}(y) \min_i \frac{c_i(y)\, r(x)}{c_i(x)} \\
&\geq \sum_{x \in S} \sum_{y \in \texttt{cov}(x)} \mathcal{D}(y) \min_i r(y) - \epsilon c_i(y) \qquad \text{(by Definition E.1(i))} \\
&= \textstyle\sum_{y \in X} \mathcal{D}(y) \min_i r(y) - \epsilon c_i(y) \\
&\geq \textstyle\sum_{y \in X} \mathcal{D}(y)\, (r(y) - \epsilon \sum_i c_i(y)) \\
&= r(\mathcal{D}) - \epsilon \textstyle\sum_i c_i(\mathcal{D}). \qquad (17)
\end{aligned}$$

Let $\tau(\mathcal{D}) = \min_i \frac{B}{c_i(\mathcal{D})}$ be the stopping time in the linear relaxation, so that $\texttt{LP}(\mathcal{D}, \mu) = \tau(\mathcal{D})\, r(\mathcal{D})$. By Equation (16) we have $\tau(\mathcal{D}_S) \geq \tau(\mathcal{D})$. We are ready for the final computation:

$$\begin{aligned}
\texttt{LP}(\mathcal{D}_S, \mu) &= \tau(\mathcal{D}_S)\, r(\mathcal{D}_S) \\
&\geq \tau(\mathcal{D})\, r(\mathcal{D}_S) \\
&\geq \tau(\mathcal{D})\, (r(\mathcal{D}) - \epsilon \textstyle\sum_i c_i(\mathcal{D})) \qquad \text{(by Equation (17))} \\
&\geq r(\mathcal{D})\, \tau(\mathcal{D}) - \epsilon\, \tau(\mathcal{D}) \textstyle\sum_i c_i(\mathcal{D}) \\
&\geq \texttt{LP}(\mathcal{D}, \mu) - \epsilon\, d\, B. \qquad \square
\end{aligned}$$

E.1 Discretization for dynamic procurement

A little more work is needed to apply uniform discretization for dynamic procurement. Consider dynamic procurement with non-unit supply. Throughout this subsection, assume n agents, budget B, and ceiling Λ. Recall that in each round t the algorithm makes an offer of the form (p_t, λ_t), where $p_t \in [0, 1]$ is the posted price per unit, and $\lambda_t \leq \Lambda$ is the maximal number of units offered for sale in this round. The action space here is $X = [0, 1] \times \{1, \ldots, \Lambda\}$, the set of all possible (p, λ) pairs.

For each arm $x = (p, \lambda)$, let $S(x)$ be the expected number of items sold. Recall that expected consumption is $c(x) = p\,S(x)$, and expected reward is $S(x)$. It follows that $\frac{r(x)}{c(x)} = \frac{1}{p}$. By Definition E.1 price p ϵ-covers price $q < p$ if and only if $\frac{1}{p} \geq \frac{1}{q} - \epsilon$.

It is easy to see that the hyperbolic ϵ-mesh S on $[0,1]$ is an ϵ-discretization: namely, each arm (q, λ) is ϵ-covered by (q, λ), where p is the smallest price in S such that $p \geq q$. Unfortunately, such S has infinitely many points. In fact, it is easy to see that any ϵ-discretization on X must be infinite (even for $\Lambda = 1$). To obtain a finite ϵ-discretization, we only consider prices $p \geq p_0$, for some parameter p_0 to be tuned later. We argue that this restriction is not too damaging.

We use a refinement of a notion of LP-optimal distribution over arms, see Appendix C.

Claim E.4. *Consider dynamic procurement with non-unit supply. Then for any $p_0 \in (0,1)$ it holds that*

$$\texttt{OPT}_{\texttt{LP}}(\{0\} \cup [p_0, 1]) \geq \texttt{OPT}_{\texttt{LP}} - \Lambda^2 p_0 n^2 / B.$$

Proof. When $p_0 > B/(n\Lambda)$ the bound is trivial and for the rest of the proof we assume that $p_0 \leq B/(n\Lambda)$. Let $\mathcal{D}$ be the distribution over arms which maximizes $\texttt{LP}(\mathcal{D}, \mu)$ to get $\texttt{OPT}_{\texttt{LP}}$. As proved in Appendix C, we can further assume that $\mathcal{D}$ is LP-perfect. Thus, $\mathcal{D}$ has a support set of size at most 2, say arms $x_i = (p_i, \lambda_i)$, $i = 1, 2$, where p_i is the posted price and λ_i is the maximal allowed number of items. W.l.o.g. assume $p_1 \leq p_2$. Note that p_1 can be 0, which would correspond to the "null arm". Moreover, letting s_i denote the expected number of items bought for arm (p_i, λ_i), then by LP-perfectness the expected consumption is

$$c(\mathcal{D}, \mu) = \textstyle\sum_{i=1}^{2} \mathcal{D}(x_i)\, p_i s_i \leq B/n. \tag{18}$$

(Formally, to apply Claim C.1 one needs to divide the rewards and consumptions (and hence the budget) by Λ, so that consumptions and rewards in each round are at most 1.)

Let us define a distribution $\mathcal{D}'$ which has support in $\{0\} \cup [p_0, 1]$.

- If $p_1 \geq p_0$ then define $\mathcal{D}' = \mathcal{D}$ and we get $\texttt{OPT}_{\texttt{LP}}(\{0\} \cup [p_0, 1]) \geq \texttt{OPT}_{\texttt{LP}}$.
- From here on, assume $p_1 < p_0$. If $\mathcal{D}$ is a distribution of support 1 (i.e $\mathcal{D}'(x_2) = 0$) then define $\mathcal{D}'(B/(n\Lambda), \lambda_1) = 1$ and we get $\texttt{OPT}_{\texttt{LP}}(\{0\} \cup [p_0, 1]) \geq \texttt{OPT}_{\texttt{LP}}$.
- If $\mathcal{D}$ is a distribution of support 2 then by LP-perfectness Equation (18) is satisfied with equality. Therefore $p_2 \geq B/(ns_2) \geq B/(n\Lambda)$. Define

$$\begin{aligned}
\mathcal{D}'(p_0, \lambda_1) &= \mathcal{D}(p_1, \lambda_1) \\
\mathcal{D}'(p_2, \lambda_2) &= \max(0, \mathcal{D}(p_2, \lambda_2) - \frac{p_0 \Lambda}{p_2}) \\
\mathcal{D}'(0) &= 1 - \mathcal{D}'(p_0) - \mathcal{D}'(p_2).
\end{aligned}$$

  Then $\mathcal{D}'$ forms a feasible solution to the linear program (5), with support in $\{0\} \cup [p_0, 1]$, and with value $\texttt{LP}(\mathcal{D}', \mu) \geq \texttt{OPT}_{\texttt{LP}} - \Lambda p_0 n / p_2 \geq \texttt{OPT}_{\texttt{LP}} - \Lambda^2 p_0 n^2 / B$.

□

Let ALG be a BwK algorithm which achieves regret bound (1). Suppose ALG is applied to a discretized instance of dynamic procurement, with m arms, n agents and budget B. For unit supply ($\Lambda = 1$), ALG has regret $R(m, n, B) = \tilde{O}(\sqrt{mn} + n\sqrt{m/B})$. For non-unit supply, in order to apply ALG we need to normalize the problem instance so that per-round rewards and consumptions are at most 1. That is, we need to divide all rewards, consumptions, and the budget, by the factor of Λ. Thus we obtain regret $R(m, n, B/\Lambda)$ on the normalized instance, and therefore regret $\Lambda\, R(m, n, B/\Lambda)$ on the original problem instance. Using Lemma E.3 and Claim E.4 (and normalizing / re-normalizing) we obtain the following regret bound:

Corollary E.5. *Consider dynamic procurement with non-unit supply. Fix $\epsilon, p_0 \in (0,1)$, and let S be the hyperbolic ϵ-mesh on $[p_0, 1]$. Then running ALG on arms $S \times \{1, \ldots, \Lambda\}$ achieves regret*

$$\Lambda\, R(\Lambda |S|,\, n, B/\Lambda) + \epsilon B + p_0\, \Lambda^2\, n^2 / B.$$

Optimizing the parameters ϵ, p_0 in Corollary E.5, we obtain the final regret bound.

Theorem E.6. *Consider dynamic procurement with non-unit supply. Assume n agents, budget B and ceiling Λ. Let ALG be a BwK algorithm which achieves regret bound (1). Applying ALG with a suitably chosen discretization yields regret $\tilde{O}(\Lambda^{3/2}\, n / B^{1/4})$.*