2013 IEEE 54th Annual Symposium on Foundations of Computer Science

Bandits with Knapsacks (Extended Abstract)

Ashwinkumar Badanidiyuru, Robert Kleinberg
Cornell University, Computer Science Department
Ithaca, NY 14850, USA
Email: {ashwin85,rdk}@cs.cornell.edu

Aleksandrs Slivkins
Microsoft Research Silicon Valley
Mountain View, CA 94043, USA
Email: slivkins@microsoft.com

***Abstract*—Multi-armed bandit problems are the predominant theoretical model of exploration-exploitation tradeoffs in learning, and they have countless applications ranging from medical trials, to communication networks, to Web search and advertising. In many of these application domains the learner may be constrained by one or more supply (or budget) limits, in addition to the customary limitation on the time horizon. The literature lacks a general model encompassing these sorts of problems. We introduce such a model, called "bandits with knapsacks", that combines aspects of stochastic integer programming with online learning. 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.**

We present two algorithms whose reward is close to the information-theoretic optimum: one is based on a novel "balanced exploration" paradigm, while the other is a primal-dual algorithm that uses multiplicative updates. Further, we prove that the regret achieved by both algorithms is optimal up to polylogarithmic factors. We illustrate the generality of the problem by presenting applications in a number of different domains including electronic commerce, routing, and scheduling. As one example of a concrete application, we consider the problem of dynamic posted pricing with limited supply and obtain the first algorithm whose regret, with respect to the optimal dynamic policy, is sublinear in the supply.

***Keywords*-Multi-armed bandits, exploration-exploitation tradeoff, regret, stochastic packing, dynamic pricing, dynamic procurement, dynamic ad allocation.**

I. Introduction

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 [1]. 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.

A common feature in many of these application domains is the presence of one or more 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. In fact, we prove that this convergence takes place at the information-theoretically optimal rate.

Problem description: bandits with knapsacks. 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, but we will also consider extensions with an infinite set of arms, see Sections V and VI.) 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. [1] The objective is to maximize the total reward received before time τ.

Full version is available at `arxiv.org/abs/1305.2545`. This extended abstract omits all proofs and many of the applications.
Parts of this research have been done while A. Badanidiyuru was a research intern at Microsoft Research Silicon Valley, while R. Kleinberg was a Consulting Researcher at Microsoft Research Silicon Valley, and while A. Slivkins was visiting Microsoft Research New York. A. Badanidiyuru was partially supported by NSF grant IIS-0905467. R. Kleinberg was partially supported by NSF grants CCF-0643934, IIS-0905467 and AF-0910940, a Microsoft Research New Faculty Fellowship, and a Google Research Grant.

[1]More generally we could model the budget constraint as a downward-closed polytope rather than a box constraint, see the full version.

The conventional MAB problem, with a finite time horizon T, naturally fits into this framework. There is a single resource called "time", one unit of which is deterministically consumed in each decision period, and the budget is T. A more interesting example is the *dynamic pricing* problem faced by a retailer selling k items to a population of T unit-demand consumers who arrive sequentially. Modeling this as a `BwK` problem, the arms correspond to the possible prices which may be offered to a consumer. Resource consumption vectors express the number of items sold and consumers seen. Thus, if a price p is offered and accepted, the reward is p and the resource consumption is $\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]$. If the offer is declined, the reward is 0 and the resource consumption is $\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$.

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 problems in optimal control have long been known to be PSPACE-hard [2].

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 the full version for examples) highlights a qualitative difference between the `BwK` problem and conventional MAB problems,[2] and it is one reason why the former problem is significantly more difficult to solve.

Our results and techniques. 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 a regret bound greater than k is worthless.

We instead seek regret bounds that are sublinear in $\texttt{OPT}$, the expected reward of the optimal policy, or (at least) sublinear in $M_{\texttt{LP}}$, the maximal possible value of $\texttt{OPT}$ 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 $\texttt{OPT}$ as both $\texttt{OPT}$ and B tend to infinity. More precisely, denoting the number of arms by m, our algorithm's regret is

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

In particular, if all budget constraints (including the time horizon) are scaled up by the factor of $\alpha > 1$, regret scales as $\sqrt{\alpha}$. The algorithm is computationally efficient, in a strong sense: the per-round running time is polynomial in the number of arms and the number of resources. We also present another algorithm (called `BalanceBwK`) whose regret bound is qualitatively similar; instead of depending on $\texttt{OPT}$ it depends on $M_{\texttt{LP}}$. The two algorithms use quite different techniques.

Algorithm `BalanceBwK` explicitly optimizes over mixtures of arms, based on a very simple idea: balanced exploration inside confidence bounds. The design principle underlying confidence-bound based algorithms for stochastic MAB (including the famous `UCB1` algorithm [7] and our algorithm `PD-BwK`) is generally, "Exploit as much as possible, but use confidence bounds that are wide enough to encourage some exploration." Our algorithm's design principle, in contrast, could be summarized as, "Explore as much as possible, but use confidence bounds that are narrow enough to eliminate obviously suboptimal alternatives." In fact, `BalanceBwK` balances its exploration across arms, so that (essentially) *each arm* is explored as much as possible. More precisely, for each arm, there are designated rounds when, once the obviously suboptimal mixtures of arms are eliminated, the algorithm picks a mixture that approximately maximizes the probability of choosing this arm. Intriguingly, the algorithm matches the logarithmic regret bound of `UCB1` for stochastic MAB (up to constant factors) and achieves qualitatively similar bounds to `PD-BwK` for `BwK`, despite being based on a design principle that is the polar opposite of those algorithms. We believe that the "balanced exploration" principle underlying `BalanceBwK` is a novel and important conceptual contribution to the theory of MAB algorithms that is likely to find other applications.

Algorithm `PD-BwK` is a primal-dual algorithm based on the multiplicative weights update method. It maintains a vec-

[2]Algorithms for explore-exploit learning problems do not typically improve over the performance of the best fixed action, with a few notable exceptions in [3], [4], [5], [6].

tor 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 [3] or the weighted majority [8] and Hedge [9] 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.

Further, we provide a matching lower bound: we prove that regret (1) is optimal up to polylog factors. Specifically, we show that any algorithm for BwK must incur regret

$$\Omega\left(\min\left(\mathtt{OPT},\ \mathtt{OPT}\sqrt{m/B}+\sqrt{m\,\mathtt{OPT}}\right)\right), \qquad (2)$$

in the worst-case over all instances of BwK with given $(m, B, \mathtt{OPT})$. We derive this lower bound using a simple example in which all arms have reward 1 and 0-1 consumption of a single resource, and one arm has slightly smaller expected resource consumption than the rest. To analyze this example, we apply the KL-divergence technique from the MAB lower bound in [3]. Some technical difficulties arise (compared to the derivation in [3]) because the arms are different in terms of the expected consumption rather than expected reward, and because we need to match the desired value for OPT.

Applications and special cases. Due to its generality, the BwK problem admits applications in diverse domains such as dynamic pricing, dynamic procurement, ad allocation, repeated auctions, network routing, and scheduling. These applications are discussed in Section VI; below we provide some highlights.

The BwK setting subsumes dynamic pricing with limited supply, as studied in [10], [11], [4] (and [12], [13] for the special case of unlimited supply). Specializing our regret bounds to this setting, we obtain the optimal regret $\tilde{O}(k^{2/3})$ with respect to the optimal policy, where k is the number of items. The prior work [11] achieved the same regret bound with respect to the best fixed price, which is a much weaker benchmark, and proved a matching lower bound. Further, our setting allows to incorporate a number of generalizations such as selling multiple products (with a limited supply of each), volume pricing, pricing bundles of goods, and profiling of buyers according to their types. In particular, we improve over the result in [4] for multiple products.

A "dual" problem to dynamic pricing is *dynamic procurement* [5], [14], where the algorithm is "dynamically buying" rather than "dynamically selling". (The budget constraint now applies to the amount spent rather than the quantity of items sold, which is why the problems are not merely identical up to sign reversal.) This problem is particularly relevant to the emerging domain of *crowdsourcing*: the items "bought" then correspond to microtasks ordered on a crowdsourcing platform such as Amazon Turk. We obtain significant improvements for the basic version of dynamic procurement studied in [5], [14]. In particular, [5] achieves a constant-factor approximation to the optimum with a prohibitively large constant (at least in the tens of thousands), whereas our approximation factor $1+\tilde{O}((T/\mathtt{OPT})B^{-1/4})$ is typically much smaller. The regret bound in [14] is against the best-fixed-price benchmark, which may be much smaller than OPT (the actual regret bounds are incomparable). Furthermore, the generality of BwK allows to incorporate generalizations such as multiple types of items/microtasks, and the presence of competing algorithms.

Further, BwK applies to the problem of dynamic ad allocation, in the context of pay-per-click advertising with unknown click probabilities. It allows to extend a standard (albeit idealized) model of ad allocation to incorporate advertisers' budgets. In fact, advertisers can specify multiple budget constraints on possibly overlapping subsets of ads.

Discussion. If the BwK instance includes multiple, different budget constraints, our regret bounds are with respect to the *smallest* of these constraints. This, however, is inevitable for the worst-case regret bounds.

Algorithm BalanceBwK is *domain-aware*, in the sense that it explicitly optimizes over all latent distributions that are feasible for a given BwK domain (such as dynamic pricing with limited supply). We do not provide a computationally efficient implementation for this optimization. In fact, it is not clear what model of computation would be appropriate to characterize access to the domain knowledge. Efficient implementation of BalanceBwK may be possible for some specific BwK domains, although we do not pursue that direction in this paper. (Recall that PD−BwK, on the other hand, is very computationally efficient.)

It is worth noting that our regret bounds for BalanceBwK and PD−BwK are incomparable: while PD−BwK achieves a stronger general bound, BalanceBwK performs better on some important special cases by virtue of being domain-aware. Specifically, it leads to a stronger regret bound of $\tilde{O}(\sqrt{k})$ for dynamic pricing with supply k under a monotone hazard rate assumption.

Open questions. While our regret bound for PD−BwK is optimal up to logarithmic factors, improved results may be possible for various special cases, especially ones involving discretization over a multi-dimensional action space.

The study of multi-armed bandit problems with large strategy sets has been a very fruitful line of investigation. It seems likely that some of the techniques introduced here could be wedded with the techniques from that literature. In particular, it would be intriguing to try combining our primal-dual algorithm `PD-BwK` with confidence-ellipsoid algorithms for stochastic linear optimization (e.g. see [15]), or enhancing the `BalanceBwK` algorithm with the technique of adaptively refined discretization, as in the zooming algortihm of [16].

It is tempting to ask about the *adversarial* version of `BwK`. However, achieving sublinear regret bounds for such version appears hopeless even for fixed-arm benchmark. In order to make progress in the positive direction, one may require a more subtle notion of benchmark, and perhaps also some restrictions on the power of the adversary.

Related work. The study of prior-free algorithms for stochastic MAB problems was initiated by [17], [7]. Subsequent work supplied algorithms for stochastic MAB problems in which the set of arms can be infinite and the payoff function is linear, concave, or Lipschitz-continuous; see a recent survey [18] for more background. Confidence bound techniques have been an integral part of this line of work, 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 [19] gave a constant-factor approximation algorithm for the Bayesian case of this problem, which was later generalized by Gupta et al. [20] to settings in which the arms' reward processes need not be martingales. Tran-Thanh et al. [21], [22], [23] presented prior-free algorithms for this problem; the best such algorithm achieves a regret guarantee qualitatively similar to that of the `UCB1` algorithm.

As discussed above, several recent papers studied dynamic pricing with limited supply [10], [11], [4] and dynamic procurement on a budget [5], [14] which, in hindsight, can be cast as special cases of `BwK` featuring a two-dimensional resource constraint. Another special case of `BwK` (discussed in more detail in the full version) has been studied in [24].

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 [25], 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 [26], [27], [28], [29]. 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 [27], 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. Moreover, our `BalanceBwK` algorithm manages to achieve strong regret guarantees without even computing a dual solution.

Map of the paper. This extended abstract focuses on stating the algorithms and the main results, with a necessary amount of preliminaries. Also, we discuss the main applications of `BwK`. The full version, available on `arxiv.org`, contains a lot of additional material, including all proofs, a section on the lower bound, a section on discretization of the action space, and a more detailed discussion of the applications.

II. 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 $(r_t; c_{t,1}, c_{t,2}\,,\; \dots\;, c_{t,d}) \in [0,1]^{d+1}$ 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 $\texttt{OPT}$. Note that $\texttt{OPT}$ 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 $\texttt{BwK}$ 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 $\texttt{BwK}$ 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. The BwK domain induces a set of feasible latent structures, which we denote $\mathcal{M}_{\texttt{feas}}$. For notational convenience, we will write $\mu_x = (\, r(x,\mu);\, c_1(x,\mu)\,,\, \dots\, , c_d(x,\mu)\,)$. Also, we will write the expected consumption as a vector $c(x,\mu) = (\, c_1(x,\mu)\,,\, \dots\, , c_d(x,\mu)\,)$. If $\mathcal{D}$ is a distribution over arms, let $r(\mathcal{D},\mu) = \sum_{x\in X} \mathcal{D}(x)\, r(x,\mu)$ and $c(\mathcal{D},\mu) = \sum_{x\in X} \mathcal{D}(x)\, c(x,\mu)$ be, respectively, the expected reward and expected resource consumption in a single round if an arm is sampled from distribution $\mathcal{D}$.

High-probability events. We will use the following expression, which we call the *confidence radius*.

$$\texttt{rad}(\nu,N) = \sqrt{C_{\texttt{rad}}\,\nu/N} + C_{\texttt{rad}}/N. \qquad (3)$$

Here $C_{\texttt{rad}} = \Theta(\log(d\,T|X|))$ is a parameter which we fix later; we will keep it implicit in the notation. The meaning of Equation (3) and $C_{\texttt{rad}}$ is explained by the following tail inequality from [16], [11].

Theorem II.1 ([16], [11])**.** *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*

$$\Pr\left[\, |\nu-\widehat{\nu}| \leq \texttt{rad}(\widehat{\nu},N) \leq 3\,\texttt{rad}(\nu,N)\,\right] \geq 1 - e^{-\Omega(C_{\texttt{rad}})}, \quad \textit{for each } C_{\texttt{rad}} > 0. \qquad (4)$$

If the expectation ν is a latent quantity, Equation (4) allows us to estimate ν with a high-confidence interval $[\widehat{\nu} - \texttt{rad}(\widehat{\nu},N),\;\; \widehat{\nu} + \texttt{rad}(\widehat{\nu},N)]$, whose endpoints are observable (known to the algorithm). Note that our estimate becomes sharper for small ν. In the full version, we use a stronger (martingale) version of Theorem II.1.

III. 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 $\texttt{BwK}$ 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 $\texttt{BwK}$.

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

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

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

Lemma III.1. $\texttt{OPT}_{\texttt{LP}}$ *is an upper bound on the value of the optimal dynamic policy.*

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:

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

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

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

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 $D^* = \xi/\,\|\xi\|_1$.

Claim III.2. *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.)*

IV. Main contributions

Our main contribution is a pair of algorithms for solving the BwK problem: a "balanced exploration" algorithm (BalanceBwK) that prioritizes information acquisition, exploring each arm as frequently as possible given current confidence intervals, and a primal-dual algorithm (PD-BwK) that chooses arms to greedily maximize the estimated reward per unit of resource cost. The two algorithms are complementary. While their regret guarantees have the same worst-case dependence on the budget parameter B, the primal-dual algorithm achieves better worst-case dependence on d and OPT. On the other hand, the balanced exploration algorithm, being more flexible, is better able to take advantage of domain-specific side information to improve its regret guarantee in favorable instances. For example, when applied to the dynamic pricing problem with supply k and with monotone hazard rate distributions, BalanceBwK achieves $\widetilde{O}(k^{1/2})$ regret whereas the primal-dual approach achieves $\widetilde{O}(k^{2/3})$ regret, just as it does for worst-case distributions.

A. Balanced exploration: algorithm BalanceBwK

The design principle behind BalanceBwK is to explore as much as possible while avoiding obviously suboptimal strategies. On a high level, the algorithm is very simple. The goal is to converge on an LP-perfect distribution. The time is divided in phases of $|X|$ rounds each. In the beginning of each phase p, the algorithm prunes away all distributions $\mathcal{D}$ over arms that with high confidence are not LP-perfect given the observations so far. The remaining distributions over arms are called *potentially perfect*. Throughout the phase, the algorithm chooses among the potentially perfect distributions. Specifically, for each arm x, the algorithm chooses a potentially perfect distribution $\mathcal{D}_{p,x}$ which approximately maximizes $\mathcal{D}_{p,x}(x)$, and "pulls" an arm sampled independently from this distribution. This choice of $\mathcal{D}_{p,x}$ is crucial; we call it the *balancing step*. The algorithm halts as soon as the time horizon is met, or any of the constraints is exhausted. See Algorithm 1 for the pseudocode.

Algorithm 1 BalanceBwK

```
1: For each phase $p = 0, 1, 2, \ldots$ do
2:     Recompute the set $\mathbf{\Delta}_p$ of potentially perfect
           distributions $\mathcal{D}$ over arms.
3:     Over the next $|X|$ rounds, for each $x \in X$:
4:         pick any distribution $\mathcal{D} = \mathcal{D}_{p,x} \in \mathbf{\Delta}_p$ such that
               $\mathcal{D}(x) \geq \frac{1}{2} \max_{\mathcal{D} \in \mathbf{\Delta}_p} \mathcal{D}(x)$.
5:         choose an arm as an independent sample from $\mathcal{D}$.
6:         halt if time horizon is met or one of the resources
           is exhausted.
```

We believe that BalanceBwK, like UCB1 [7], is a very general design principle and has the potential to be a meta-algorithm for solving stochastic online learning problems.

Theorem IV.1. *Consider an instance of BwK with d resources, $m = |X|$ arms, and the smallest budget $B = \min_i B_i$. Let $M_{\mathtt{LP}}$ be the maximum of $\mathtt{OPT}_{\mathtt{LP}}$, for given time horizon and budgets, over all problem instances in a given BwK domain. For algorithm BalanceBwK, the total expected regret $\mathtt{OPT} - \mathtt{REW}$ is at most*

$$O(\log(dmT)\log(T/m)) \left(\sqrt{dm\,M_{\mathtt{LP}}} + M_{\mathtt{LP}}\left(\sqrt{dm/B} + dm/B\right) + dm\right). \quad (7)$$

Let us fill in the details in the specification of BalanceBwK. In the beginning of each phase p, the algorithm recomputes a "confidence interval" I_p for the latent structure μ, so that (informally) $\mu \in I_p$ with high probability. Then the algorithm determines which distributions $\mathcal{D}$ over arms can potentially be LP-perfect given that $\mu \in I_p$. Specifically, let $\mathbf{\Delta}_p$ be set of all distributions $\mathcal{D}$ that are LP-perfect for some latent structure $\mu' \in I_p$; such distributions are called *potentially perfect* (for phase p).

It remains to define the confidence intervals I_p. For phase $p = 0$, the confidence interval I_0 is simply $\mathcal{M}_{\mathtt{feas}}$, the set of all feasible latent structures. For each subsequent phase $p \geq 1$, the confidence interval I_p is defined as follows. For each arm x, consider all rounds before phase p in which this arm has been chosen. Let $N_p(x)$ be the number of such rounds, let $\widehat{r}_p(x)$ be the time-averaged reward in these rounds, and let $\widehat{c}_{p,i}(x)$ be the time-averaged consumption of resource i in these rounds. We use these averages to estimate $r(x,\mu)$ and $c_i(x,\mu)$ as follows:

$$|r(x,\mu) - \widehat{r}_p(x)| \leq \mathtt{rad}\left(\widehat{r}_p(x), N_p(x)\right) \quad (8)$$

$$|c_i(x,\mu) - \widehat{c}_{p,i}(x)| \leq \mathtt{rad}\left(\widehat{c}_{p,i}(x), N_p(x)\right) \quad (\forall\, i) \quad (9)$$

The confidence interval I_p is the set of all latent structures $\mu' \in I_{p-1}$ that are consistent with these estimates. This completes the specification of BalanceBwK.

For each phase of BalanceBwK, the round in which an arm is sampled from distribution $\mathcal{D}_{p,x}$ will be called *designated* to arm x. We need to use approximate (rather than exact) maximization to choose $\mathcal{D}_{p,x}$ because an exact maximizer $\operatorname{argmax}_{\mathcal{D} \in \mathbf{\Delta}_p} \mathcal{D}(x)$ is not guaranteed to exist.

B. 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. To give 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 [30], [31], [32], 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 learning algorithms, it is far from obvious that this particular hybrid of the two methods should be effective. In particular, the use of multiplicative updates on dual variables, rather than primal ones, distinguishes our algorithm from other bandit algorithms that use multiplicative weights (e.g. the Exp3 algorithm [3]) and brings it closer in spirit to the literature on stochastic packing algorithms (especially [27]).

Algorithm 2 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 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 $\operatorname{argmin}_{z \in \mathbf{\Delta}[X]} \{ \frac{y_t^\intercal L_t z}{u_t^\intercal z} \}$.
9:     $v_{t+1} = \operatorname{Diag}\{(1+\epsilon)^{\mathbf{e}_j^\intercal L_t z_t}\} v_t$.
```

The pseudocode for the algorithm is presented as Algorithm 2, which we call PD-BwK. In the pseudocode, we 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 $c_i(x_j, \mu)$.

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 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* $[0,1] \cap [\hat{\nu} - \mathtt{rad}(\hat{\nu}, N),\ \hat{\nu} + \mathtt{rad}(\hat{\nu}, N)]$. The UCB or LCB for a vector or matrix are defined componentwise.

In step 8, the pseudocode asserts that the set $\operatorname{argmin}_{z \in \mathbf{\Delta}[X]} \{ \frac{y_t^\intercal L_t z}{u_t^\intercal z} \}$ contains at least one of the point-mass distributions $\{\mathbf{e}_1, \ldots, \mathbf{e}_m\}$. This is true, because if $\rho = \min_{z \in \mathbf{\Delta}[X]} \{ (y_t^\intercal L_t z)/(u_t^\intercal z) \}$ then the linear inequality $y_t^\intercal L_t z \le \rho u_t^\intercal \mathcal{D}$ holds at some point $z \in \mathbf{\Delta}[X]$, 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 [31]. 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.)

Theorem IV.2. *For any instance of the* BwK *problem, Algorithm* PD-BwK *satisfies*

$$\mathtt{OPT} - \mathtt{REW} \le O\left(\sqrt{\log(dmT)}\right) \left(\sqrt{m\,\mathtt{OPT}} + \mathtt{OPT}\sqrt{m/B} + m\sqrt{\log(dmT)}\right). \quad (10)$$

Comparing this bound to Theorem IV.1, when $d = O(1)$ and $M_{\text{LP}} = O(\mathtt{OPT})$ the two guarantees are the same up to logarithmic factors. The regret guarantee for the primal-dual algorithm scales better with the number of resources, d, and it can be vastly superior in cases where $\mathtt{OPT} \ll M_{\text{LP}}$.

V. BwK WITH DISCRETIZATION

In many applications of BwK the action space X is very large or infinite, so the algorithms developed in the previous sections are not immediately applicable. However, in these applications the action space has some structure that our algorithms can leverage. For example, in dynamic pricing (with one type of good) every possible action (arm) corresponds to a price – i.e., a number in some fixed interval.

To handle such applications, we use a simple approach called *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. In the dynamic pricing application, S can consist of all prices in the allowed interval that are of the form $\ell\epsilon$, $\ell \in \mathbb{N}$, for some fixed discretization parameter $\epsilon > 0$. We call this subset the *additive ϵ-mesh*. The granularity of discretization (i.e., the value of ϵ) is adjusted in advance so as to minimize regret.

This generic approach has been successfully used in past work on dynamic pricing (e.g., [13], [12], [10], [11]) to provide worst-case optimal regret bounds. To carry it through to the setting of BwK, we need to define an appropriate notion of discretization (which would generalize the additive ϵ-mesh for dynamic pricing), 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. We flesh out the details in the full version.

For dynamic pricing, we show that if we use the additive ϵ-mesh S_ϵ, and $R(S_\epsilon)$ is the regret on the problem instance restricted to S_ϵ, then regret on the original problem instance is at most $\epsilon\, d B + R(S_\epsilon)$, where d is the number of constraints. More generally, we prove this for any BwK domain, and any subset $S_\epsilon \subset X$ which satisfies some axioms for a given parameter ϵ; we call such S_ϵ an *ϵ-discretization*. Once we define an ϵ-discretization S_ϵ for every $\epsilon > 0$, then in order to optimize the regret bound $\epsilon\, d B + R(S_\epsilon)$ up to constant factors it suffices to pick ϵ such that $\epsilon\, dB = R(S_\epsilon)$. We need to consider regret bounds $R(S_\epsilon)$ that are in terms of the maximal possible expected total reward $M_{\texttt{LP}}$ rather than the (possibly smaller) OPT, since the value of OPT is not initially known to the algorithm.

VI. Applications

Owing to its generality, the BwK problem admits applications in many different domains. Below we describe three applications: to dynamic pricing, dynamic procurement and crowdsourcing, and dynamic ad allocation. More applications (to repeated auctions, network routing, and dynamic scheduling) are discussed in the full version.

A. Dynamic pricing with limited supply

In the basic version, the algorithm is a seller which has k identical items for sale and is facing n agents (potential buyers) that are arriving sequentially. Each agent is interested in buying one item. The algorithm offers a take-it-or-leave-it price p_t to each arriving agent t. The agent has a fixed value $v_t \in [0, 1]$ for an item, which is *private*: not known to the algorithm. The agent buys an item if and only if $p_t \geq v_t$. We assume that each v_t is an independent sample from some fixed (but unknown) distribution, called the buyers' *demand distribution*. The goal of the algorithm is to maximize his expected revenue.

This problem has been studied in [13], [12], [10], [11]. The best result is an algorithm that achieves regret $\tilde{O}(k^{2/3})$ compared to the best fixed price [11] (the $\tilde{O}(\cdot)$ notation hides poly-log factors). This regret rate is proved to be worst-case optimal. For demand distributions that satisfy a standard (but limiting) assumption of *regularity*, the best fixed price is essentially as good as the best dynamic policy.

Dynamic pricing is naturally interpreted as a BwK domain: arms correspond to prices, rounds correspond to arriving agents (so there is a time horizon of n), and there is a single type of resource: the k items. One can think of the agents as simply units of time, so that n is the time horizon. Let $S(p)$ be the probability of a sale at price p. The latent structure μ is determined by the function $S(\cdot)$: the expected reward is $r(p, \mu) = p\, S(p)$, and expected resource consumption is simply $c(x, \mu) = S(p)$. Since there are only k items for sale, it follows that OPT $\leq k$. (To analyze BalanceBwK, note that the LP-values are bounded above by k, so we can take $M_{\texttt{LP}} = k$.) Prices can be discretized using the ϵ-uniform mesh over $[0, 1]$, resulting in $\frac{1}{\epsilon}$ arms.

We recover the optimal $\tilde{O}(k^{2/3})$ regret result from [11] with respect to offering the same price to each buyer, and moreover extend this result to regret with respect to the optimal dynamic policy. We observe that the expected revenue of the optimal dynamic policy can be twice the expected revenue of the best fixed price (see the full version). The crucial technical advance compared to the prior work is that our algorithms strive to converge to an optimal *distribution* over prices, whereas the algorithms in prior work target the best fixed price. Technically, the result follows by applying the "quality guarantee" for the additive ϵ-mesh (fleshed out in the full version), in conjunction with either of the two main algorithms, and optimizing the ϵ.

B. Dynamic procurement on a budget

A "dual" problem to dynamic pricing is *dynamic procurement*, where the algorithm is buying rather than selling. 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 seller's value for an item is an independent sample from some fixed (but unknown) distribution with support $[0, 1]$. The algorithm offers a take-it-or-leave-it price to each arriving agent. The goal is to maximize the number of items bought. As discussed in the introduction, this problem has been studied in [5], [14].

Application to crowdsourcing. 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 extensions.

Dynamic procurement via BwK. Let us cast dynamic procurement as a BwK domain. As in dynamic pricing, agents correspond to time rounds, and n is the time horizon. The only resource is money. Arms correspond to prices: $X = [0, 1]$. Letting $S(p)$ be the probability of a sale at price p, the expected reward (items bought) is $r(p, \mu) = S(p)$, and the expected resource consumption is $c(p, \mu) = pS(p)$. The best a-priori upper bound on the LP-value is $M_{\texttt{LP}} = n$.

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 an additive δ-mesh 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 significantly fewer arms. Optimizing the parameters p_0 and ϵ, we obtain regret $\tilde{O}\left(n/B^{1/4}\right)$; see the full version for details.

Our result is meaningful, and improves over the constant-factor approximation in [5], as long as OPT is not too small compared to $n/B^{1/4}$. Recall that our result is with respect to the optimal dynamic policy. We observe that in this domain the optimal dynamic policy can be vastly superior compared to the best fixed price (see the full version for an example).

C. Extensions for dynamic pricing/procurement

In the full version we discuss several extensions of the basic dynamic pricing/procurement models outlined above, including: non-unit demands/supplies, multiple products to sell/buy, bundling and volume pricing, and competitive environment. As an example, we outline a specific setting: dynamic pricing with multiple products.

The algorithm has d products for sale, with a limited inventory of each. In each round t, an agent arrives. The agent is characterized by the vector of values $(v_{t,1}, \ldots, v_{t,d}) \in [0, 1]^d$, one for each product i. This vector is private: not known to the algorithm. We assume that it is an independent sample from a fixed but unknown distribution over $[0, 1]^d$; note that arbitrary correlations between values of different products are allowed. The algorithm offers a vector of prices $(p_{t,1}, \ldots, p_{t,d}) \in [0, 1]^d$, one for each product i. The agent buys one unit of each product i such that $p_{t,i} \geq v_{t,i}$.

D. Dynamic ad allocation with unknown click probabilities

Consider *pay-per-click* (PPC) advertising on the web (in particular, this is a prevalent model in sponsored search auctions). The central premise in PPC advertising is that an advertiser derives value from her ad only when the user clicks on this ad. The ad platform allocates ads to users that arrive over time.

Consider the following simple (albeit highly idealized) model for PPC ad allocation. Users arrive over time, and the ad platform needs to allocate an ad to each arriving user. There is a set X of available ads. Each ad x is characterized by the payment-per-click π_x and click probability μ_x; the former quantity is known to the algorithm, whereas the latter is not. If an ad x is chosen, it is clicked on with probability μ_x, in which case payment π_x is received. The goal is to maximize the total payment. This setting, and various extensions thereof that incorporate user/webpage context, has received a considerable attention in the past several years (starting with [33]). In fact, the connection to PPC advertising has been one of the main motivations for the recent surge of interest in MAB.

We enrich the above setting by incorporating advertisers' *budgets*. In the most basic version, for each ad x there is a budget B_x — the maximal amount of money that can be spent on this ad. More generally, an advertiser can have an ad campaign which consists of a subset S of ads, so that there is a per-campaign budget S. Even more generally, an advertiser can have a more complicated budget structure: a family of overlapping subsets $S \subset X$ and a separate budget B_S for each S. For example, BestBuy can have a total budget for the ad campaign, and also separate budgets for ads about TVs and ads about computers. Finally, in addition to budgets (i.e., constraints on the number of times ads are clicked), an advertiser may wish to have similar constraints on the number of times ads are shown. BwK allows us to express all these constraints.

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 to routing protocols generalizes a network routing problem that was communicated to us by Luyi Gui. The application of dynamic procurement to crowdsourcing have been suggested to us by Chien-Ju Ho and Jennifer Wortman Vaughan.

References

[1] P. Whittle, "Multi-armed bandits and the Gittins index," *J. Royal Statistical Society, Series B*, vol. 42, no. 2, pp. 143–149, 1980.

[2] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of optimal queuing network control," *Math. Oper. Res.*, vol. 24, no. 2, pp. 293–305, 1999.

[3] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The nonstochastic multiarmed bandit problem." *SIAM J. Comput.*, vol. 32, no. 1, pp. 48–77, 2002, preliminary version in *36th IEEE FOCS*, 1995.

[4] O. Besbes and A. J. Zeevi, "Blind network revenue management," *Operations Research*, vol. 60, no. 6, pp. 1537–1550, 2012.

[5] A. Badanidiyuru, R. Kleinberg, and Y. Singer, "Learning on a budget: posted price mechanisms for online procurement," in *13th ACM EC*, 2012, pp. 128–145.

[6] I. Abraham, O. Alonso, V. Kandylas, and A. Slivkins, "Adaptive crowdsourcing algorithms for the bandit survey problem," in *26th COLT*, 2013.

[7] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem." *Machine Learning*, vol. 47, no. 2-3, pp. 235–256, 2002, preliminary version in *15th ICML*, 1998.

[8] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," *Information and Computation*, vol. 108, no. 2, pp. 212–260, 1994.

[9] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[10] O. Besbes and A. Zeevi, "Dynamic pricing without knowing the demand function: Risk bounds and near-optimal algorithms," *Operations Research*, vol. 57, pp. 1407–1420, 2009.

[11] M. Babaioff, S. Dughmi, R. Kleinberg, and A. Slivkins, "Dynamic pricing with limited supply," in *13th ACM EC*, 2012.

[12] A. Blum, V. Kumar, A. Rudra, and F. Wu, "Online learning in online auctions," in *14th ACM-SIAM SODA*, 2003, pp. 202–204.

[13] R. Kleinberg and T. Leighton, "The value of knowing a demand curve: Bounds on regret for online posted-price auctions." in *44th IEEE FOCS*, 2003, pp. 594–605.

[14] A. Singla and A. Krause, "Truthful incentives in crowdsourcing tasks using regret minimization mechanisms," in *22nd WWW*, 2013, pp. 1167–1178.

[15] V. Dani, T. P. Hayes, and S. Kakade, "Stochastic Linear Optimization under Bandit Feedback," in *21th COLT*, 2008, pp. 355–366.

[16] R. Kleinberg, A. Slivkins, and E. Upfal, "Multi-Armed Bandits in Metric Spaces," in *40th ACM STOC*, 2008, pp. 681–690.

[17] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocations rules," *Adv. in Appl. Math.*, vol. 6, pp. 4–22, 1985.

[18] S. Bubeck and N. Cesa-Bianchi, "Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems," *Foundations and Trends in Machine Learning*, vol. 5, no. 1, pp. 1–122, 2012.

[19] S. Guha and K. Munagala, "Multi-armed Bandits with Metric Switching Costs," in *36th ICALP*, 2007, pp. 496–507.

[20] A. Gupta, R. Krishnaswamy, M. Molinaro, and R. Ravi, "Approximation algorithms for correlated knapsacks and non-martingale bandits," in *52nd IEEE FOCS*, 2011, pp. 827–836.

[21] L. Tran-Thanh, "Budget-limited multi-armed bandits," Ph.D. dissertation, University of Southampton, 2012.

[22] L. Tran-Thanh, A. Chapman, E. M. de Cote, A. Rogers, and N. R. Jennings, "ϵ-first policies for budget-limited multi-armed bandits," in *24th AAAI*, 2010, pp. 1211–1216.

[23] L. Tran-Thanh, A. Chapman, A. Rogers, and N. R. Jennings, "Knapsack based optimal policies for budget-limited multi-armed bandits," in *26th AAAI*, 2012, pp. 1134–1140.

[24] N. Cesa-Bianchi, C. Gentile, and Y. Mansour, "Regret minimization for reserve prices in second-price auctions," in *ACM-SIAM SODA*, 2013.

[25] N. R. Devanur and T. P. Hayes, "The AdWords problem: Online keyword matching with budgeted bidders under random permutations," in *10th ACM EC*, 2009, pp. 71–78.

[26] S. Agrawal, Z. Wang, and Y. Ye, "A dynamic near-optimal algorithm for online linear programming," 2009, technical report. Available from arXiv at `http://arxiv.org/abs/0911.2974`.

[27] N. R. Devanur, K. Jain, B. Sivan, and C. A. Wilkens, "Near optimal online algorithms and fast approximation algorithms for resource allocation problems," in *12th ACM EC*, 2011, pp. 29–38.

[28] J. Feldman, M. Henzinger, N. Korula, V. S. Mirrokni, and C. Stein, "Online stochastic packing applied to display ad allocation," in *18th ESA*, 2010, pp. 182–194.

[29] M. Molinaro and R. Ravi, "Geometry of online packing linear programs," in *39th ICALP*, 2012, pp. 701–713.

[30] S. Arora, E. Hazan, and S. Kale, "The multiplicative weights update method: a meta-algorithm and applications," *Theory of Computing*, vol. 8, no. 1, pp. 121–164, 2012.

[31] N. Garg and J. Könemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," *SIAM J. Computing*, vol. 37, no. 2, pp. 630–652, 2007.

[32] S. A. Plotkin, D. B. Shmoys, and E. Tardos, "Fast approximation algorithms for fractional packing and covering problems," *Mathematics of Operations Research*, vol. 20, pp. 257–301, 1995.

[33] S. Pandey, D. Agarwal, D. Chakrabarti, and V. Josifovski, "Bandits for Taxonomies: A Model-based Approach," in *SDM*, 2007.