Efficient Contextual Bandits with Knapsacks via Regression
(Technical Report)

Aleksandrs Slivkins*       Dylan Foster*

November 2022

Abstract

We consider contextual bandits with knapsacks (CBwK), a variant of the contextual bandit which places global constraints on budget consumption. We present a new algorithm that is simple, statistically optimal, and computationally efficient. Our algorithm combines LagrangeBwK [19], a Lagrangian-based technique for CBwK, and SquareCB [11], a regression-based technique for contextual bandits. Our analysis emphasizes the modularity of both techniques.

1 Introduction

In contextual bandits with knapsacks (CBwK), an algorithm sequentially chooses among a fixed set of arms and consumes $d$ constrained resources. In each round, an algorithm observes a context, chooses an arm, receives a reward, and also consumes some amount of each resource. Thus, the outcome of choosing an arm is now a $(d + 1)$-dimensional vector rather than a scalar. The algorithm stops when the total consumption of some resource $i$ exceeds its respective budget. CBwK is a common generalization of two well-studied bandit problems: contextual bandits (the special case with no resources) and bandits with knapsacks (BwK, the special case without contexts).

There are essentially two main approaches to handle contextual bandits: one posits a classification oracle [2, 3], the other posits a regression oracle and assumes (approximate) realizability [12, 11, 25]. We follow the latter approach, because it allows for a much more computationally efficient algorithm for CBwK, and is known to be (at least) as good as the best classification-based approaches in contextual bandit experiments. Specifically, we build on the SquareCB algorithm from Foster and Rakhlin [11].

On the BwK side, we build the LagrangeBwK algorithm of Immorlica, Sankararaman, Schapire, and Slivkins [19]. This algorithm is structured as a repeated zero-sum game between the “primal” player which chooses among the arms, and the “dual” player which chooses among the resources, with payoffs given by a natural Lagrangian relaxation. The analysis of LagrangeBwK is modular in the sense that it admits any application-specific primal algorithm with a particular regret guarantee.

Our presentation emphasizes the modularity inherent in both techniques. In particular, we incorporate their respective analyses as specific theorems, and re-use much of the technical setup. Our contribution here is to assemble the pieces and identify how they connect to one another. As a result, the analysis becomes quite simple and very transparent.

*Microsoft Research NYC. Emails: {slivkins,dylanfoster}@microsoft.com
Background. The problem of bandits with knapsacks was introduced and optimally solved in Badanidiyuru et al. [8]. A long list of motivating examples and a detailed survey of the literature can be found in Chapter 11 of Slivkins [26]. Agrawal and Devanur [4] and Immorlica et al. [19] provide alternative regret-optimal algorithms. In particular, the algorithm in [4], which we call UCB-BwK, implements “optimism under uncertainty” paradigm and underlies some CBwK algorithms in prior work. Our approach to CBwK is based on yet another approach from Immorlica et al. [19].

The study of CBwK was initiated in Badanidiyuru et al. [7] and continued in [5, 3, 19, 24]. In particular, Agrawal et al. [5] consider CBwK with classification oracles and obtain a statistically optimal and theoretically oracle-efficient algorithm which combines UCB-BwK and the “minimonster” technique for contextual bandits [2]. Agrawal and Devanur [3] solve a linear version of CBwK, combining UCB-BwK and the optimistic approach for linear contextual bandits [22, 9, 1]; the same algorithm and analysis (and a number of extensions) are recovered via a generic reduction in Sankararaman and Slivkins [24].

Compared to stochastic bandits, BwK is more challenging in several ways. (1) Resource consumption during exploration may limit the algorithm’s ability to exploit in the future rounds. A stark consequence is that Explore-first algorithm fails if the budgets are too small (see Exercise 10.1(a) in Slivkins [26]). (2) Per-round expected reward is no longer the right objective. An arm with high per-round expected reward may be undesirable because of high resource consumption. Instead, one needs to think about the total expected reward over the entire time horizon. (3) Learning the best arm is no longer the right objective. Instead, one is interested in the best fixed distribution over arms. This is because a fixed distribution over arms can perform much better than the best fixed arm. All three challenges arise in the “basic” special case when there is only one resource (other than the time itself), only two arms, and the budgets are linear in the time horizon.

The name “bandits with knapsacks” comes from an analogy with the well-known knapsack problem in algorithms. In that problem, one has a knapsack of limited size, and multiple items each of which has a value and takes a space in the knapsack. The goal is to assemble the knapsack: choose a subset of items that fits in the knapsacks so as to maximize the total value of these items. However, in BwK the “value” and “size” of a given action are not known in advance.

Contextual bandits and bandits with knapsacks are variants / generalizations of multi-armed bandits, a paradigmatic model for exploration-exploitation tradeoff and a subject of thousands of papers and several books. The most recent books are Lattimore and Szepesvári [21] and Slivkins [26].

Concurrent work. A paper on the same problem, Han et al. [17], appeared on arxiv.org very recently. Han et al. [17] obtain a similar result using an algorithm similar to ours. The main technical difference is that they do not explicitly express their algorithm as an instantiation of LagrangeBwK, and accordingly do not take advantage of the modularity therein. We emphasize that our work is simultaneous and independent with respect to Han et al. [17]. In particular, a preliminary draft of this technical report (without proofs) was circulated informally in Spring’22.

2 Problem formulation and preliminaries

The problem is defined as follows. There are $K$ actions (a.k.a. arms) and $d$ constrained resources being consumed by the algorithm. One of these resources is time: each arm consumes one unit of the “time resource” in each round, and its budget is the time horizon $T$. The algorithm stops when the total consumption of some resource $i$ exceeds its respective budget $B_i$. 


In each round $t$, an algorithm observes a context $x_t \in \mathcal{X}$ from some set $\mathcal{X}$ of possible contexts, chooses an arm $a_t \in [K]$, receives a reward $r_t$, and also consumes some amount $c_{t,i}$ of each resource $i$. Thus, the outcome of choosing an arm is now a $(d+1)$-dimensional vector rather than a scalar. As a technical assumption, the reward and consumption of each resource in each round lie in $[0, 1]$.

| Problem protocol: Contextual Bandits with Knapsacks (CBwK) |
|-------------------------------------------------------------|
| Parameters: $K$ arms, $T$ rounds, context space $\mathcal{X}$, budgets $B_1, \ldots, B_d \in [0, T]$. For each round $t \in [T]$:
| 1. Algorithm observes a context $x_t \in \mathcal{X}$ and chooses an arm $a_t \in [K]$. |
| 2. Outcome vector $\tilde{\sigma}_t = (r_t; c_{t,1}, \ldots, c_{t,d}) \in [0, 1]^{d+1}$ is observed, |
|   where $r_t$ is the algorithm’s reward, and $c_{t,i}$ is consumption for each resource $i$. |
| Algorithm stops when the total consumption of some resource $i$ exceeds its budget $B_i$. |

We focus on an i.i.d. variant of this problem. In particular, we assume:

- *i.i.d. contexts*: in each round $t$, the context $x_t$ is drawn independently from a fixed distribution over $\mathcal{X}$.
- *i.i.d. outcomes*: for each context and each arm, the outcome vector is sampled independently from a fixed distribution over outcome vectors.

Let $B = \min_{i \in [d]} B_i$ be the smallest budget. Without loss of generality, we rescale the problem so that all budgets are $B$: we divide the per-round consumption of each resource $i$ by $B_i/B$. In particular, the per-round consumption of the time resource is now $B/T$. We focus on the paradigmatic regime $B = \Omega(T)$. Formally, we treat $\rho := B/T$ as a constant parameter relative to the time horizon $T$ (but keep the dependence on $\rho$ explicit).

Formally, an instance of CBwK is specified by parameters $T, B, K, d$, a distribution over contexts, and a mapping from context-arm pairs to distributions over outcome vectors. The algorithm’s goal is to maximize the total reward, denoted $\text{Rew}(\text{ALG})$.

Following the literature on BwK, we consider regret relative to the expected total reward of the best algorithm for a particular problem instance:

$$\text{Opt} := \sup_{\text{algorithms } \text{ALG}} \mathbb{E}[\text{Rew}(\text{ALG})].$$

**Remark 2.1.** CBwK subsumes “contextual bandits” and “bandits with knapsacks” (BwK), as the special cases when, resp., there are no budgets (i.e., $B = T$) and there are no contexts (i.e., $|\mathcal{X}| = 1$).

**Remark 2.2.** While most analyses in contextual bandits seamlessly carry over to adversarial context arrivals, this is not the case for CBwK. Indeed, algorithms for CBwK with adversarial context arrivals does not admit sublinear regret, and instead are doomed to a constant approximation ratio.

We posit that one of the arms, called the *null arm*, brings no reward and consumes no resource except the “time resource”. Playing this arm is tantamount to skipping a round. The null arm is

---

1Strictly speaking, the reward in very last round should not count; we ignore this point for simplicity.

2To see this, consider a version of a simple “spend or save” dilemma from [10]. There are three types of contexts, deterministically yielding high, low, and medium rewards, respectively. The contexts are “medium” in the first half of the rounds, and either all “high” or all “low” in the second half. The algorithm would not know whether to spend all its budget in the first half, or save it for the second half.
essential for the analysis: it ensures that anything (any arm, policy, or distribution over policies) can be scaled down so as to satisfy the budget constraints.

Remark 2.3. Even when skipping rounds is not allowed, existence of the null arm comes without loss of generality. Indeed, whenever the null arm is chosen, an algorithm can proceed to the next round internally, without actually outputting an action. After $T$ rounds have passed from the algorithm’s perspective, the algorithm can choose arms arbitrarily.

Regression oracles. We posit the existence of a regression oracle, following [12, 11, 25]. Let’s define what it means for rewards; for consumption of a given resource, it is defined similarly. A regression function is a function $X \times [K] \to [0,1]$ which predicts expected rewards for a given context-arm pair. Let $\mathcal{F}_{\text{all}}$ denote the class of all regression functions. A regression oracle for a given function class $F \subset \mathcal{F}_{\text{all}}$ is a subroutine that inputs the history of algorithm’s rewards, i.e., a collection of data points of the form $(x_t, a_t, r_t)$, and outputs a regression function from $\mathcal{F}$ which approximates the (true) expected rewards. The underlying intuition is that the regression oracle is computationally efficient. We focus on an online regression oracle that processes the data points one by one; the intuition is that the update time does not depend on the time horizon.

More formally, an online regression oracle is an algorithm for the online regression problem. We define this problem below using more abstract notation (which we subsequently use in the analysis).

Problem protocol: Online regression with oracle $O$

Parameters: $K$ arms, $T$ rounds, context space $\mathcal{Y}$, range $[a, b] \subset \mathbb{R}$.

In each round $t \in [T]$:

1. the oracle outputs a regression function $f_t \in \mathcal{F}_{\text{all}}[a, b] := \{ f : \mathcal{Y} \times [K] \to [a, b] \}$.
2. Adversary chooses context $y_t \in \mathcal{Y}$, arm $a_t \in [K]$, score $z_t \in [a, b]$, and auxiliary data $\text{aux}_t$ (if any).
3. the oracle inputs the new data-point $(y_t, a_t, z_t, \text{aux}_t)$.

Implicitly, $f_t(x_t, a_t)$ estimates the expected score $\mathbb{E}[z_t \mid x_t, a_t]$ for each round $t$.

For simplicity, we posit the same online regression oracle for rewards and for each resource. However, our algorithm and analysis can easily accommodate a different oracle (possibly with a different function class) for each component of the outcome vector.

Additional notation. The sets of rounds, arms and resources are denoted $[T]$, $[K]$ and $[d]$, respectively. The distributions over arms and resources are denoted $\Delta_K$ and $\Delta_d$, respectively.

For each arm $a$, context $x$ and resource $i$, the expected reward and resource consumption is

$$r(x, a) = \mathbb{E} \left[ r_t(a) \mid x_t = x \right] \text{ and } c_i(x, a) = \mathbb{E} \left[ c_{t,i}(a) \mid x_t = x \right]. \quad (2.1)$$

A policy is a deterministic mapping from contexts to arms. The set of all policies is denoted $\Pi_{\text{all}}$.

Fix some distribution $D$ over policies. Suppose this distribution is “played” in a given round $t$, i.e., a policy $\pi$ is drawn independently from $D$, and then an arm is chosen according to $\pi$: $a_t = \pi(x_t)$. The expected reward and resource-i consumption are defined as

$$r(D) = \mathbb{E}_{\pi \sim D} \left[ r_t(\pi(x_t)) \right] \text{ and } c_i(D) = \mathbb{E}_{\pi \sim D} \left[ c_{t,i}(\pi(x_t)) \right]. \quad (2.2)$$
Linear program and Lagrangian payoffs. A standard linear relaxation of CBwK is defined as follows. The intuition is that an algorithm for CBwK implicitly chooses a policy in each round, and would like to converge on the best distribution over policies. Therefore, we optimize over distributions $D$ over policies, setting the per-round reward and resource- $i$ consumption to be deterministically equal to $r(D)$ and $c_i(D)$, respectively. We are only interested in distributions $D$ such that the algorithm does not run out of resources until round $T$. Thus, we have a linear program:

$$\begin{align*}
\text{maximize} & \quad r(D) \\
\text{subject to} & \quad D \in \Delta_{\Pi_{\text{all}}} \\
& \quad T \cdot c_i(D) \leq B \quad \forall i \in [d].
\end{align*}$$

(2.3)

The value of this linear program is denoted $\text{Opt}_{\text{LP}}$. One can prove that the corresponding total reward, $T \cdot \text{Opt}_{\text{LP}}$, is an upper bound on the best expected reward achievable in BwK.

Consider the Lagrange function associated with the linear program (2.3). For our purposes, this function inputs a distribution $D$ over policies and a distribution $\lambda$ over resources,

$$L(D, \lambda) := r(D) + \sum_{i \in [d]} \lambda_i \left[1 - \frac{T}{B} c_i(D)\right].$$

(2.4)

Consider the expected Lagrangian game: a zero-sum game between two players, the primal player that chooses arms and the dual player that chooses resources, with payoffs given by (2.4) and interpreted as rewards for the primal player, and costs for the dual player. This game is related to the LP as follows:

**Lemma 2.4** ([19]). Suppose $(D^*, \lambda^*)$ is a mixed Nash equilibrium for the Lagrangian game. Then $L(D^*, \lambda^*) = \text{Opt}_{\text{LP}}$, and $D^*$ is an optimal solution for the linear program (2.3).

3 Algorithmic framework and results

We define an algorithm for CBwK called LagrangeCBwK. We use a version of LagrangeBwK framework from Immorlica et al. [19]. Here, we have a repeated zero-sum game between two algorithms: $\text{Alg}_{\text{Prim}}$ that chooses arms, and $\text{Alg}_{\text{Dual}}$ that chooses distributions over resources.\footnote{The terms ‘primal’ and ‘dual’ here refer to the duality in linear programming. For the LP-relaxation (2.3), primal variables correspond to arms, and dual variables (i.e., variables in the dual LP) correspond to resources.} The round-$t$ payoff for choosing arm $a$ and a distribution $\lambda$ over resources is given by

$$L_t(a, \lambda) = r_t(a) + 1 - \frac{T}{B} \sum_{i \in [d]} \lambda_i \cdot c_{t,i}(a).$$

(3.1)

This payoff constitutes reward for $\text{Alg}_{\text{Prim}}$, and cost for $\text{Alg}_{\text{Dual}}$. Note that $E[L_t(a, \lambda)] = L(a, \lambda)$. Note that $\text{Alg}_{\text{Dual}}$ moves first, and $\text{Alg}_{\text{Prim}}$ responds to the chosen $\lambda_t$. This is crucial for our algorithm, whereas in Immorlica et al. [19] the players move simultaneously. $\text{Alg}_{\text{Dual}}$ could observe the context $x_t$, but it does not seem to help. Note that $\text{Alg}_{\text{Dual}}$ receives full feedback on its Lagrange costs: indeed, the outcome vector $\tilde{o}_t$ allows to reconstruct $L_t(a_t,i)$ for each resource $i \in [d]$. For our purposes, $\text{Alg}_{\text{Dual}}$ could be any algorithm for online learning with full feedback, such as the exponential weights method (or, “Hedge”) [15].

The intuition behind LagrangeBwK is as follows. If $\text{Alg}_{\text{Prim}}$ and $\text{Alg}_{\text{Dual}}$ satisfy certain regret-minimizing properties, the repeated game converges to the Nash equilibrium. This holds for any definition of $L_t(a, \lambda)$ that is linear in $\lambda$. The specific definition (3.1) ensures that the strategy of $\text{Alg}_{\text{Prim}}$ in the Nash equilibrium is essentially optimal for the CBwK instance, as per Lemma 2.4.
The novelty in LagrangeCBwK is defining \( \text{Alg}_{\text{Prim}} \), based on SquareCB, a regression-based technique for contextual bandits from Foster and Rakhlin [11]. \( \text{Alg}_{\text{Prim}} \) is parameterized by an online regression oracle, denoted \( \text{Alg}_{\text{Est}} \), which is applied separately to rewards and to each resource. The pseudocode is given in Algorithm 2. In particular, Eq. (3.2) is a natural way to estimate expected Lagrange payoffs given the regression functions. Eq. (3.3) is the “secret sauce” in SquareCB: the formula which makes it work.

The per-round running time of \( \text{Alg}_{\text{Prim}} \) is dominated by \( d+1 \) oracle calls and \( K(d+1) \) evaluations of the regression functions \( \hat{f}_t \) in (3.2). For the probabilities in (3.3), it takes \( O(K) \) time to compute the max expressions, and then \( O(K \log \frac{1}{\epsilon}) \) time to binary-search for \( c_t^{\text{norm}} \) up to a given accuracy \( \epsilon \).

It is instructive (and essential for the analysis) to formally realize \( \text{Alg}_{\text{Prim}} \) as an instantiation of...
Define the \textit{Lagrange regression} as an online regression problem with data points of the form \((y_t, a_t, z_t, \text{aux}_t)\) for each round \(t\), where the context \(y_t = (x_t, \lambda_t)\) consists of both the CBwK context \(x_t\) and the dual vector \(\lambda_t\), the score \(z_t = L_t(a_t, \lambda_t)\) is the Lagrangian payoff as defined by Eq. (3.1), and the auxiliary data \(\text{aux}_t = \tilde{a}_t\) is the outcome vector. The \textit{Lagrange oracle} \(\mathcal{O}_{\text{Lag}}\) is an algorithm for this problem (i.e., an online regression oracle) which, for each round \(t\), uses the estimated Lagrangian payoff (3.1) as a regression function. Thus, \(\text{Alg}_{\text{Prim}}\) is an instantiation of \text{SquareCB} for Lagrange regression, with oracle \(\mathcal{O}_{\text{Lag}}\).

### 3.2 Provable guarantees

To state our guarantees, let us formalize the online regression problem faced by a given oracle \(\mathcal{O}_i\) in the algorithm, for each \(i \in [d + 1]\); call it \text{EstProblem}_i. In each round \(t\) of this problem, the context \(x_t\) is drawn as in CBwK, i.e., from a random source of contexts, and the arm \(a_t\) is chosen arbitrarily, possibly depending on the history. The score is \(z_t = (\tilde{a}_t)_i\), the \(i\)-th component of the outcome vector for the \((x_t, a_t)\) pair. Let \(f_t^*\) be the correct regression function for \text{EstProblem}_i, i.e., a regression function \(f_t^* \in \mathcal{F}\) all given by

\[
    f_t^*(x, a) = \mathbb{E}[ (\tilde{a}_t)_i | x_t = x, a_t = a ] \quad \forall x \in \mathcal{X}, a \in [K].
\]  

(3.4)

Following the literature on online regression, we consider the \textit{squared regression error}:

\[
    \text{Est}_i(\mathcal{O}_i) := \sum_{t \in [T]} \left( \hat{f}_{t,i}(x_t, a_t) - f_t^*(x_t, a_t) \right)^2, \quad \forall i \in [d + 1].
\]  

(3.5)

We rely on a known uniform high-probability upper-bound on these errors:

\[
\forall \delta \in (0, 1) \quad \exists U_\delta > 0 \quad \forall i \in [d + 1] \quad \Pr \left[ \text{Est}_i(\mathcal{O}_i) \leq U_\delta \right] \geq 1 - \delta.
\]  

(3.6)

\textbf{Theorem 3.1.} Consider algorithm \text{LagrangeCBwK} with \text{Alg}_{\text{Est}} alg that satisfies (3.7). Fix an arbitrary failure probability \(\delta \in (0, 1)\) and set the parameter \(\gamma = \frac{B}{T} \frac{\sqrt{K}}{(d + 1)U}\), where \(U = U_\delta/(d + 1)\). Let \text{Alg}_{\text{Dual}} be implemented using the exponential weights algorithm (“Hedge”) [15]. Then with probability at least \(1 - O(\delta T)\), the algorithm’s total reward \(\text{Rew}\) satisfies

\[
\text{Opt} - \text{Rew} \leq O \left( \frac{T}{\gamma} \sqrt{dTU \log(dT/\delta)} \right).
\]

To ensure that (3.6) is satisfied for a given index \(i\), one typically posits some function class \(\mathcal{F} \subset \mathcal{F}_{\text{all}}\) that satisfies realizability: \(f_t^* \in \mathcal{F}\). Then one typically obtains (3.6) with \text{Alg}_{\text{Est}} tailored to \(\mathcal{F}\) and \(U_\delta = U_0 + \log(2/\delta)\), where \(U_0 < \infty\) typically reflects the intrinsic statistical capacity of the class \(\mathcal{F}\) [27, 6, 29, 16, 23]. Standard examples include:

- Finite classes, for which Vovk’s aggregating algorithm [27] achieves \(U_0 = \mathcal{O}(\log |\mathcal{F}|)\).
- Linear classes, where

\[
\mathcal{F} = \left\{ (x, a) \mapsto \langle \theta, \phi(x, a) \rangle | \theta \in \mathbb{R}^d, \|\theta\|_2 \leq 1 \right\}
\]

for a known feature map \(\phi(x, a) \in \mathbb{R}^d\) with \(\|\phi(x, a)\|_2 \leq 1\). Here, the Vovk-Azoury-Warmuth algorithm [28, 6] achieves \(U_0 \leq \mathcal{O}(d \log(T/d))\). If \(d\) is very large, one could also use Online Gradient Descent (e.g., Hazan [18]) and achieve \(U_0 \leq \mathcal{O}(\sqrt{T})\).
We emphasize that (3.6) can also be ensured via approximate versions of realizability, with the upper bound $U_\delta$ depending on the quality of the approximation. Any such guarantee from the literature on regressions seamlessly plugs into our theorem.

We refer to Foster and Rakhlin [10] for further background.

### 3.3 Extensions

In what follows, we sketch some natural extensions to Theorem 3.1.

1. While our theorem sets parameter $\gamma$ according to the known upper bound $U_\delta$, in practice it may be advantageous to treat $\gamma$ as a hyperparameter and tune it experimentally.

2. Instead of computing (3.3) and binary-searching for the normalization constant $c_{\text{norm}}$, one can do the following (cf. Foster and Rakhlin [10]). Let $b_t = \arg\max_{a \in \{K\}} \hat{\mathcal{L}}_t(a)$. For all $a \neq b_t$, set
   \[
   p_t(a) = 1 / \left( K + \gamma \cdot (\hat{\mathcal{L}}_t(b_t) - \hat{\mathcal{L}}_t(a)) \right),
   \]
   and set $p_t(b_t) = 1 - \sum_{a \neq b_t} p_t(a)$. This attains the same regret bound as Algorithm 2 (see Theorem 4.2), up to a factor of 2.

3. SquareCB allows for various extensions to large, structured action sets, some of which are practical. Any such extension should carry over to CBwK and LagrangeCBwK. Essentially, one needs to efficiently implement sampling from distribution in (3.3). “Practical” extensions are known for action sets with linear structure [13, 30], and those with Lipschitz-continuity structure (via uniform discretization) [14]. More extensions to general action spaces, RL, and beyond, at varying levels of practicality, can be found in [14].

4. In practice, one could potentially implement the Lagrange oracle by applying AlgEst to the entire Lagrange payoffs $\mathcal{L}_t(a_t, \lambda_t)$ directly, with $(x_t, \lambda_t)$ as a context.

5. In some applications, the outcome vector is determined by an observable “fundamental outcome” of lower dimension. For example, in dynamic pricing an algorithm offers an item for sale at a given price $p$, and the “fundamental outcome” is whether there is a sale. The corresponding outcome vector is $(p, 1) \cdot 1_{\text{sale}}$, i.e., a sale brings reward $p$ and consumes 1 unit of resource. In such applications, we should apply regression directly to the fundamental outcomes.

### 4 Analysis: Proof of Theorem 3.1

For the sake of analysis, we allow our algorithm to run till round $T$ (but stop counting its reward once some resource runs out of budget, as before).

**LagrangeCBwK analysis.** We incorporate the generic analysis of LagrangeCBwK as follows. Define the primal problem (resp., dual problem) as the online learning problem faced by AlgPrim (resp., AlgDual) from the perspective of the LagrangeCBwK framework. The primal problem is a bandit problem where algorithm’s action set is the set $\Pi_{\text{all}}$ of all contextual bandit policies, with stochastic context arrivals and Lagrange payoffs as rewards. The dual problem is a full-feedback online learning problem where algorithm’s “actions” are the resources in CBwK, with Lagrange payoffs as
costs. The primal regret (resp., dual regret) is the regret relative to the best-in-hindsight action in the respective problem. Formally, we define these quantities for any given stopping time $\tau \in [T]$:

$$\text{Reg}_{\text{Prim}}(\tau) := \left( \max_{\pi \in \Pi_{\text{all}}} \sum_{t \in [\tau]} \mathcal{L}_t(\pi(x_t), \lambda_t) \right) - \sum_{t \in [\tau]} \mathcal{L}_t(a_t, \lambda_t).$$

$$\text{Reg}_{\text{Dual}}(\tau) := \sum_{t \in [\tau]} \mathcal{L}_t(a_t, \lambda_t) - \left( \max_{i \in [d]} \sum_{t \in [\tau]} \mathcal{L}_t(a_t, i) \right).$$

The guarantee in Immorlica et al. [19] assumes high-probability upper bounds on the primal and dual regret:

$$\Pr \left[ \forall \tau \in [T] \quad \text{Reg}_{\text{Prim}}(\tau) \leq \overline{\text{Reg}}_{\text{Prim}}(T) \quad \text{and} \quad \text{Reg}_{\text{Dual}}(\tau) \leq \overline{\text{Reg}}_{\text{Dual}}(T) \right] \geq 1 - \delta, \quad (4.1)$$

where $\overline{\text{Reg}}_{\text{Prim}}(T)$ and $\overline{\text{Reg}}_{\text{Dual}}(T)$ are some functions of $T$ determined by the problem instance, and $\delta \in (0, 1)$ is a given failure probability.

**Theorem 4.1** (Immorlica et al. [19]). Consider algorithm LagrangeCBwK and assume $\text{(4.1)}$ for some $\delta > 0$. Then with probability at least $1 - O(\delta)$ the algorithm’s total reward $\text{Rew}$ satisfies

$$\text{Opt} - \text{Rew} \leq \overline{\text{Reg}}_{\text{Prim}}(T) + \overline{\text{Reg}}_{\text{Dual}}(T) + O \left( \frac{T}{\delta^2} \sqrt{T \log(dT/\delta)} \right).$$

**SquareCB analysis.** We incorporate the generic analysis of SquareCB by applying it to the Lagrange oracle $\mathcal{O}_{\text{Lag}}$ defined in the previous section. The squared regression error for this oracle is given by

$$\text{Est}(\mathcal{O}_{\text{Lag}}) = \sum_{t \in [T]} \left( \hat{\mathcal{L}}_t(a_t) - \mathcal{L}(a_t, \lambda_t) \right)^2. \quad (4.2)$$

The guarantee posits a known high-probability upper-bound on these errors:

$$\forall \delta \in (0, 1) \quad \exists U_{\text{Lag}}^\delta > 0 \quad \Pr \left[ \text{Est}(\mathcal{O}_{\text{Lag}}) \leq U_{\text{Lag}}^\delta \right] \geq 1 - \delta. \quad (4.3)$$

Restated in our notation, SquareCB analysis implies the following.

**Theorem 4.2** (Foster and Rakhlin [11]). Consider algorithm LagrangeCBwK with Lagrange oracle that satisfies $\text{(4.3)}$. Fix some failure probability $\delta \in (0, 1)$ and let $U_{\text{Lag}}^\delta$ be the upper bound from $\text{(4.3)}$. Set the parameter $\gamma = \sqrt{AT/U}$. Then with probability at least $1 - O(\delta)$ we have

$$\forall \tau \in [T] \quad \text{Reg}_{\text{Prim}}(\tau) \leq O \left( \sqrt{T U \log(dT/\delta)} \right). \quad (4.4)$$

**Error analysis.** To complete the proof, it remains to derive $\text{(4.3)}$ from $\text{(3.6)}$, i.e., upper-bound the error on $\mathcal{O}_{\text{Lag}}$ using respective upper bounds on $\text{Alg}_{\text{Est}}$. Represent $\text{Est}(\mathcal{O}_{\text{Lag}})$ as

$$\text{Est}(\mathcal{O}_{\text{Lag}}) = \sum_{t \in [T]} \left( \Phi_t + \frac{T}{B} \sum_{i \in [d]} \lambda_{t,i} \Psi_{t,i} \right)^2,$$

where $\Phi_t = f_t(x_t, a_t) - r(x_t, a_t)$ and $\Psi_{t,i} = c_i(x_t, a_t) - f_{t,i+1}(x_t, a_t)$. For each round $t$,

$$\left( \Phi_t + \frac{T}{B} \sum_{i \in [d]} \lambda_{t,i} \Psi_{t,i} \right)^2 \leq 2 \Phi_t^2 + 2 \left( \sum_{i \in [d]} \lambda_{t,i} \Psi_{t,i} \right)^2 \leq 2 \Phi_t^2 + 2 \left( \sum_{i \in [d]} \lambda_{t,i} \Psi_{t,i} \right)^2.$$
where the latter inequality follows from Jensen’s inequality. Summing this up over all rounds $t$,

$$\text{Est}(O_{\text{Lag}}) \leq (T/B)^2 \sum_{i \in [d+1]} \text{Est}_i(O_i).$$

The $(T/B)^2$ scaling is due to the fact that consumption is scaled by $T/B$ in the Lagrangian, and the error is quadratic. Consequently, (4.3) holds with $U_{\text{Lag}}^\delta = (d+1)(T/B)^2 U_{\delta/(d+1)}^\delta$.

References

[1] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In 25th Advances in Neural Information Processing Systems (NIPS), pages 2312–2320, 2011.

[2] Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert Schapire. Taming the monster: A fast and simple algorithm for contextual bandits. In 31st Intl. Conf. on Machine Learning (ICML), 2014.

[3] Shipra Agrawal and Nikhil R. Devanur. Linear contextual bandits with knapsacks. In 29th Advances in Neural Information Processing Systems (NIPS), 2016.

[4] Shipra Agrawal and Nikhil R. Devanur. Bandits with global convex constraints and objective. Operations Research, 67(5):1486–1502, 2019. Preliminary version in ACM EC 2014.

[5] Shipra Agrawal, Nikhil R. Devanur, and Lihong Li. An efficient algorithm for contextual bandits with knapsacks, and an extension to concave objectives. In 29th Conf. on Learning Theory (COLT), 2016.

[6] Katy S. Azoury and Manfred K. Warmuth. Relative loss bounds for on-line density estimation with the exponential family of distributions. Machine Learning, 43(3):211–246, June 2001.

[7] Ashwinkumar Badanidiyuru, John Langford, and Aleksandrs Slivkins. Resourceful contextual bandits. In 27th Conf. on Learning Theory (COLT), 2014.

[8] Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks. J. of the ACM, 65(3):13:1–13:55, 2018. Preliminary version in FOCS 2013.

[9] Wei Chu, Lihong Li, Lev Reyzin, and Robert E. Schapire. Contextual Bandits with Linear Payoff Functions. In 14th Intl. Conf. on Artificial Intelligence and Statistics (AISTATS), 2011.

[10] Dylan J Foster and Alexander Rakhlin. Beyond UCB: Optimal and efficient contextual bandits with regression oracles. International Conference on Machine Learning (ICML), 2020.

[11] Dylan J. Foster and Alexander Rakhlin. Beyond UCB: optimal and efficient contextual bandits with regression oracles. In 37th Intl. Conf. on Machine Learning (ICML), 2020.

[12] Dylan J. Foster, Alekh Agarwal, Miroslav Dudík, Haipeng Luo, and Robert E. Schapire. Practical contextual bandits with regression oracles. In 35th Intl. Conf. on Machine Learning (ICML), pages 1534–1543, 2018.

[13] Dylan J Foster, Claudio Gentile, Mehryar Mohri, and Julian Zimmert. Adapting to misspecification in contextual bandits. Advances in Neural Information Processing Systems, 33, 2020.

[14] Dylan J Foster, Sham M Kakade, Jian Qian, and Alexander Rakhlin. The statistical complexity of interactive decision making. arXiv preprint arXiv:2112.13487, 2021.

[15] 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.

[16] S. Gerchinovitz. Sparsity regret bounds for individual sequences in online linear regression. Journal of Machine Learning Research, 14:729–769, 2013.
[17] Yuxuan Han, Jialin Zeng, Yang Wang, Yang Xiang, and Jiheng Zhang. Optimal contextual bandits with knapsacks under realizability via regression oracles, October 2022. Working paper at arxiv.org/abs/2210.11834.

[18] Elad Hazan. *Introduction to Online Convex Optimization*. Foundations and Trends in Optimization, 2016.

[19] Nicole Immorlica, Karthik Abinav Sankararaman, Robert Schapire, and Aleksandrs Slivkins. Adversarial bandits with knapsacks. *J. of the ACM*, August 2022. Preliminary version in *60th IEEE FOCS*, 2019.

[20] John Langford and Tong Zhang. The Epoch-Greedy Algorithm for Contextual Multi-armed Bandits. In *21st Advances in Neural Information Processing Systems (NIPS)*, 2007.

[21] Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, Cambridge, UK, 2020. Versions available at https://banditalgs.com/ since 2018.

[22] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *19th Intl. World Wide Web Conf. (WWW)*, 2010.

[23] Alexander Rakhlin and Karthik Sridharan. Online nonparametric regression. In *Conference on Learning Theory*, 2014.

[24] Karthik Abinav Sankararaman and Aleksandrs Slivkins. Bandits with knapsacks beyond the worst-case. In *35th Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[25] David Simchi-Levi and Yunzong Xu. Bypassing the monster: A faster and simpler optimal algorithm for contextual bandits under realizability, 2020. Working paper, available at https://arxiv.org/abs/2003.12699.

[26] Aleksandrs Slivkins. Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning*, 12(1-2):1–286, November 2019. Published with Now Publishers (Boston, MA, USA). Also available at https://arxiv.org/abs/1904.07272. Latest online revision: Jan 2022.

[27] V. Vovk. A game of prediction with expert advice. *J. Computer and System Sciences*, 56(2):153–173, 1998.

[28] Vladimir Vovk. Competitive on-line linear regression. In *NIPS ’97: Proceedings of the 1997 conference on Advances in neural information processing systems 10*, pages 364–370, Cambridge, MA, USA, 1998. MIT Press.

[29] Vladimir Vovk. Metric entropy in competitive on-line prediction. *CoRR*, abs/cs/0609045, 2006.

[30] Yinglun Zhu, Dylan J. Foster, Paul Mineiro, and John Langford. Contextual bandits in large action spaces: Made practical. In *39th Intl. Conf. on Machine Learning (ICML)*, 2021.