SIAM J. COMPUT.
Vol. 39, No. 3, pp. 1088–1106

PLAYING GAMES WITH APPROXIMATION ALGORITHMS[*]

SHAM M. KAKADE[†], ADAM TAUMAN KALAI[‡], AND KATRINA LIGETT[§]

Abstract. In an online linear optimization problem, on each period t, an online algorithm chooses $s_t \in \mathcal{S}$ from a fixed (possibly infinite) set $\mathcal{S}$ of feasible decisions. Nature (who may be adversarial) chooses a weight vector $w_t \in \mathbb{R}^n$, and the algorithm incurs cost $c(s_t, w_t)$, where c is a fixed cost function that is linear in the weight vector. In the *full-information* setting, the vector w_t is then revealed to the algorithm, and in the *bandit* setting, only the cost experienced, $c(s_t, w_t)$, is revealed. The goal of the online algorithm is to perform nearly as well as the best fixed $s \in \mathcal{S}$ in hindsight. Many repeated decision-making problems with weights fit naturally into this framework, such as online shortest-path, online traveling salesman problem (TSP), online clustering, and online weighted set cover. Previously, it was shown how to convert any efficient *exact* offline optimization algorithm for such a problem into an efficient online algorithm in both the full-information and the bandit settings, with average cost nearly as good as that of the best fixed $s \in \mathcal{S}$ in hindsight. However, in the case where the offline algorithm is an approximation algorithm with ratio $\alpha > 1$, the previous approach worked only for special types of approximation algorithms. We show how to convert any offline approximation algorithm for a linear optimization problem into a corresponding online approximation algorithm, with a polynomial blowup in runtime. If the offline algorithm has an α-approximation guarantee, then the expected cost of the online algorithm on any sequence is not much larger than α times that of the best $s \in \mathcal{S}$, where the best is chosen with the benefit of hindsight. Our main innovation is combining Zinkevich's algorithm for convex optimization with a geometric transformation that can be applied to any approximation algorithm. Standard techniques generalize the above result to the bandit setting, except that a "barycentric spanner" for the problem is also (provably) necessary as input. Our algorithm can also be viewed as a method for playing large repeated games, where one can compute only *approximate* best responses, rather than best responses.

Key words. online linear optimization, regret minimization, approximation algorithms, online algorithms

AMS subject classification. 68W01

DOI. 10.1137/070701704

1. Introduction.

In the 1950s, Hannan gave an algorithm for playing repeated two-player games against an arbitrary opponent [12]. His was one of the earliest algorithms with the *no-regret* property: against any opponent, his algorithm achieved expected performance asymptotically near that of the best single action, where the best is chosen with the benefit of hindsight. Put another way, after sufficiently many rounds, someone using his algorithm would not benefit (significantly) by being able to change his actions to any single action, even if this action could be chosen after observing the opponent's play. Kalai and Vempala [13] showed that Hannan's approach can be used to *efficiently* solve online linear optimization problems as well. Hannan's algorithm relied on the ability to find best responses to an opponent's play history. Informally speaking, Kalai and Vempala replaced this best-reply computation with an

[*]Received by the editors August 31, 2007; accepted for publication (in revised form) September 25, 2008; published electronically September 2, 2009. A preliminary version of these results appeared in *Proceedings of the 39th Annual ACM Symposium on Theory of Computation*, 2007.
http://www.siam.org/journals/sicomp/39-3/70170.html

[†]Toyota Technological Institute, Chicago, IL 60637 (sham@tti-c.org).

[‡]College of Computing, Georgia Tech, Atlanta, GA 30332 (atk@cc.gatech.edu). This author's research was supported in part by NSF award SES-527656.

[§]Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 (katrina@cs.cmu.edu). This author's research was supported in part by an AT&T Labs Graduate Fellowship and an NSF Graduate Research Fellowship.

efficient black-box optimization algorithm (the number of calls to that algorithm on a sequence of length T was $O(\sqrt{T})$ [13]). However, the above approach breaks down when one can only approximately solve the offline optimization problem efficiently or one can only compute approximate best responses. That is the focus of the present paper.

In an offline optimization problem, one must select a single decision s from a known set of decisions $\mathcal{S}$, in order to minimize a known cost function. In an offline *linear* optimization problem, a weight vector $w \in \mathbb{R}^n$ is given as input, and the cost function $c(s, w)$ is assumed to be linear in w. Many combinatorial optimization problems fit into this framework, including traveling salesman problems (TSPs) (where $\mathcal{S}$ consists of a subset of paths in a graph), clustering ($\mathcal{S}$ is partitions of a graph), weighted set cover ($\mathcal{S}$ is the set of covers), and knapsack ($\mathcal{S}$ is the set of feasible sets of items and weights correspond to item valuations).

Each of these problems has an *online* sequential version, in which on every period the player must select her decision without knowing that period's cost function. That is, there is an unknown sequence of weight vectors $w_1, w_2, \ldots \in \mathbb{R}^n$ and for each $t = 1, 2, \ldots$, the player must select $s_t \in \mathcal{S}$ and pay $c(s_t, w_t)$. In the *full-information* version, the player is then informed of w_t, while in the *bandit* version she is only informed of the value $c(s_t, w_t)$. (The name *bandit* refers to the similarity to the classic multiarmed bandit problem [15]).

The player's goal is to achieve low average cost. In particular, we compare her cost with that of the best fixed decision: she would like her average cost to approach that of the best single point in $\mathcal{S}$, where the best is chosen with the benefit of hindsight. This difference, $\frac{1}{T}\sum_{t=1}^{T} c(s_t, w_t) - \min_{s\in\mathcal{S}} \frac{1}{T}\sum_{t=1}^{T} c(s, w_t)$, is termed *regret*.

Prior work showed how to convert an *exact* algorithm for the offline problem into an online algorithm with low regret, both in the full-information setting and in the bandit setting. In particular, Kalai and Vempala showed [13] that using Hannan's approach [12], one can guarantee $O(T^{-1/2})$ regret for any linear optimization problem, in the full-information version, as the number of periods T increases. It was later shown [2, 14, 7] how to convert exact algorithms to achieve $O(T^{-1/3})$ regret in the more difficult bandit setting.

This prior work was actually a reduction showing that one can solve the online problem *nearly as efficiently* as one can solve the offline problem. (They used the offline optimizer as a black box.) However, in many cases of interest, such as online combinatorial auction problems [4], even the offline problem is NP-hard. Hannan's "follow-the-perturbed-leader" approach can also be applied to some special types of approximation algorithms, but fails to work directly in general. Finding a reduction that maintains good asymptotic performance using *general* approximation algorithms was posed as an open problem [13]; we resolve this problem.

In this paper, we show how to convert *any* approximation algorithm for a linear optimization problem into an algorithm for the online sequential version of the problem, both in the full-information setting and in the bandit setting. Our reduction maintains the asymptotic approximation guarantee of the original algorithm, relative to the average performance of the best static decision in hindsight. Our new approach is inspired by Zinkevich's algorithm for the problem of minimizing convex functions over a convex feasible set $\mathcal{S} \subseteq \mathbb{R}^n$ [16]. However, the application is not direct and requires a geometric transformation that can be applied to any approximation algorithm.

Example 1 (online metric TSP). Every day, a delivery company serves the same n customers. The company must schedule its daily route without foreknowledge of the traffic on each street. The time on any street may vary unpredictably from day to day due to traffic, construction, accidents, or even competing delivery companies. In *online metric TSP*, we are given an undirected graph G, and on every period t, we must output a tour that starts at a specified vertex, visits all the vertices at least once, and then returns to the initial vertex. After we announce our tour, the traffic patterns are revealed (in the full-information setting, the costs on all the edges; in the bandit setting, just the cost of the tour) and we pay the cost of the tour.

Example 2 (online weighted set cover). Every financial quarter, our company hires vendors from a fixed pool of subcontractors to cover a fixed set of tasks. Each subcontractor can handle a known, fixed subset of the tasks, but their price is only announced at the end of the quarter and varies from quarter to quarter. In *online weighted set cover*, the vendors are fixed sets $P_1, \dots, P_n \subseteq [m]$. Each period, we choose a legal cover $s_t \subseteq [n]$, that is, $\bigcup_{i \in s_t} P_i = [m]$. There is an unknown sequence of cost vectors $w_1, w_2, \dots \in [0,1]^n$, indicating the quarterly vendor costs. Each quarter, our total cost $c(s_t, w_t)$ is the sum of the costs of the vendors we chose for that quarter. In the full-information setting, at the end of the quarter we find out the price charged by each of the subcontractors; in the bandit setting, we receive a combined bill showing only our total cost.

1.1. Hannan's approach. In this section, we briefly describe the previous approach [13] for the case of exact optimization algorithms based on Hannan's idea of adding perturbations. We begin with the obvious "follow-the-leader" algorithm which, on each period, picks the decision that is best against the total (equivalently, average) of the previous weight vectors. This means, on period t, choosing $s_t = A\big(\sum_{\tau=1}^{t-1} w_\tau\big)$, where A is an algorithm that, given a cost vector w, produces the best $s \in \mathcal{S}$.[1] Hannan's perturbation idea, in our context, suggests using $s_t = A\big(p_t + \sum_{\tau=1}^{t-1} w_\tau\big)$ for uniformly random perturbation $p_t \in [0, \sqrt{t}]^n$. One can bound the expected regret of following-the-perturbed-leader to be $O(T^{-1/2})$, disregarding other parameters of the problem.

Kalai and Vempala [13] note that Hannan's approach maintains an asymptotic α-approximation guarantee when used with α-approximation algorithms with a special property they call *α-pointwise approximation*, meaning that on any input, the solution they find differs from the optimal solution by a factor of at most α in every coordinate. They observe that a number of algorithms, such as the Goemans–Williamson max-cut algorithm [11], have this property. Balcan and Blum [4] observe that the previous approach applies to another type of approximation algorithm: one that uses an optimal decision for another linear optimization problem, for example, using a minimum spanning tree (MST) for TSP. It is also not difficult to see that an FPTAS (fully polynomial time approximation scheme) can be used to get a $(1+\epsilon)$-competitive online algorithm. We further note that the Hannan–Kalai–Vempala approach extends to approximation algorithms that perform a simple type of randomized rounding where the randomness does not depend on the input.

In the appendix, we use an explicit example based on the greedy set-cover approximation algorithm to illustrate how Hannan's approach fails on more general approximation algorithms.

1.2. Informal statement of results. The main result of this paper is a general conversion from any approximate linear optimization algorithm to an approximate

[1]This approach fails even on a two-decision problem, where the costs of the two decisions are (0.5,0) during the first period and then alternate $(1,0), (0,1), (1,0), \dots$, thereafter.

online version in the full-information setting (section 3). The extension to the bandit setting (section 4) uses well-understood techniques, modulo one new issue that arises in the case of approximation algorithms. We summarize the problem, our approach, and our results here.

We assume there is a known compact convex set $\mathcal{W} \subseteq \mathbb{R}^n$ of legal weight vectors (in many cases $\mathcal{W} = [0,1]^n$), and a cost function $c : \mathcal{S} \times \mathcal{W} \to [0,1]$ that is *linear* in its second argument, that is, $c(s, av + bw) = ac(s,v) + bc(s,v)$ for all $s \in \mathcal{S}$, $a, b \in \mathbb{R}$, and $v, w, av + bw \in \mathcal{W}$. The generalization to $[0, M]$-bounded cost functions for $M > 0$ is straightforward.[2] We assume that we have a black-box α-approximation algorithm, which we abstract as an oracle A such that, for all $w \in \mathcal{W}$, $c(A(w), w) \le \alpha \min_{s\in\mathcal{S}} c(s, w)$. That is, we do not assume that our approximation oracle can optimize in every direction. In the full-information setting, we assume our only access to $\mathcal{S}$ is via the approximation algorithm; in the bandit setting, we need an additional assumption, which we describe below.

In this paper, we focus on the *nonadaptive setting*, in which the adversary's choices of w_t can be arbitrary but must be chosen in advance. In the *adaptive setting*, on period t, the adversary may choose w_t based on $s_1, w_1, \ldots, s_{t-1}, w_{t-1}$. In the bandit case, extension of these results to the adaptive setting and the conversion from results in expectation to high probability results remain open questions.

For α-approximation algorithms, it is natural to consider the following notion of *α-regret*, in both the full-information and the bandit settings. It is the difference between the algorithm's average cost and α times the cost of the best $s \in \mathcal{S}$, that is, $\frac{1}{T}\sum_{t=1}^{T} c(s_t, w_t) - \alpha \min_{s\in\mathcal{S}} \frac{1}{T}\sum_{t=1}^{T} c(s, w_t)$.[3]

1.2.1. Full-information results. Our approach to the full-information problem is inspired by Zinkevich's algorithm (for a somewhat different problem) [16], which uses an exact projection oracle to create an online algorithm with low regret. An exact projection oracle Π_J is an algorithm which can produce $\operatorname{argmin}_{x\in J} ||x - y||$ for all $y \in \mathbb{R}^n$, where J is the "feasible region" (in Zinkevich's setting, it is a compact convex subset of $\mathbb{R}^n$). The main algorithm presented in Zinkevich's paper, GREEDY PROJECTION, determines its decision x_t at time t as $x_t = \Pi_J(x_{t-1} - \eta w_{t-1})$, where η is a parameter called the learning rate and w_{t-1} is the cost vector at time $(t-1)$. One can view the approach in this paper as providing a method to simulate a type of "approximate" projection oracle using an approximation algorithm. In section 3 we show the following result.

RESULT 1.1. *Given any α-approximation oracle to an offline linear optimization problem and any $T, T_0 \ge 1$, $w_1, w_2, \ldots \in \mathcal{W}$, our (full-information) algorithm outputs $s_1, s_2, \ldots \in \mathcal{S}$ achieving*

$$\mathrm{E}\left[\frac{1}{T}\sum_{t=T_0+1}^{T_0+T} c(s_t, w_t)\right] - \alpha \min_{s\in\mathcal{S}} \frac{1}{T}\sum_{t=T_0+1}^{T_0+T} c(s, w_t) = \frac{O(\alpha n)}{\sqrt{T}}.$$

The algorithm makes $\mathrm{poly}(n, T)$ *calls to the approximation oracle.*

Note that the above bound on expected α-regret holds simultaneously for every window of T consecutive periods (T must be known by the algorithm). We easily inherit this useful adaptation property of Zinkevich's algorithm. It is not clear to us whether one could elegantly achieve this property using the previous approach.

[2]In [13], the set $\mathcal{W} = \{w \in \mathbb{R}^n \mid |w|_1 \le 1\}$ was assumed.

[3]If there is a hardness of approximation result with ratio α for the offline version of a problem, one cannot expect to obtain better than α-regret efficiently in the online setting.

1.2.2. Bandit results. Previous work in the bandit setting constructs an "exploration basis" to allow the algorithm to discover better decisions [2, 14, 7]. In particular, Awerbuch and Kleinberg [2] introduce a so-called barycentric spanner (BS) as their exploration basis and show how to construct one from an optimization oracle $A : \mathbb{R}^n \to \mathcal{S}$. However, in the case where the oracle (exact or approximate) accepts inputs only in, say, the positive orthant, it may be impossible to extract an exploration basis. Hence, we assume that we are given a β-BS ($\beta \geq 1$ is an approximation factor for the BS) for the problem at hand as part of the input. Note that the β-BS needs only to be computed once for a particular problem and then can be reused for all future instances of that problem. Given a β-BS, the standard reduction from the bandit setting to the full-information setting gives the following result.

RESULT 1.2. *For any β-BS and any α-approximation oracle to an offline linear optimization problem and any $T, T_0 \geq 1$, $w_1, w_2, \ldots \in \mathcal{W}$, the (bandit) algorithm in Figure* 4.1 *outputs $s_1, s_2, \ldots \in \mathcal{S}$ achieving*

$$\mathrm{E}\left[\frac{1}{T}\sum_{t=T_0+1}^{T_0+T} c(s_t, w_t)\right] - \alpha \min_{s\in\mathcal{S}} \frac{1}{T}\sum_{t=T_0+1}^{T_0+T} c(s, w_t) = \frac{O(n(\alpha\beta)^{2/3})}{\sqrt[3]{T}}.$$

The algorithm makes $\mathrm{poly}(n, T)$ *calls to the approximation oracle.*

We also show, in section 4.1, that the assumption of a BS is necessary.

RESULT 1.3. *There is no polynomial-time black-box reduction from an α-approximation algorithm for a general linear optimization problem (without additional input) to a bandit algorithm guaranteeing low α-regret.*

We note that the above regret is suboptimal in terms of the T dependence. Furthermore, recent work [8, 3, 1] presents algorithms for online linear optimization that achieve the optimal $\sqrt{T}$ regret even in the bandit setting (these results either do not explicitly consider the computational issues or assume access to an exact optimization oracle). Achieving improved regret for bandit algorithms using approximation oracles remains an open problem.

2. Formal definitions.

We formalize the natural notion of an n-dimensional linear optimization problem.

DEFINITION 2.1. *An n-*dimensional linear optimization problem *consists of a convex compact set of* feasible weight vectors *$\mathcal{W} \subset \mathbb{R}^n$, a set of* feasible decisions *$\mathcal{S}$, and a cost function $c : \mathcal{S} \times \mathcal{W} \to [0, 1]$ that is linear in its second argument.*

Due to the linearity of c, there must exist a mapping $\Phi : \mathcal{S} \to \mathbb{R}^n$ such that $c(s, w) = \Phi(s) \cdot w$ for all $s \in \mathcal{S}, w \in \mathcal{W}$. In the case where the standard basis is contained in $\mathcal{W}$, we have

$$\Phi(s) = \big(c(s, (1, 0, \ldots, 0)), \ldots, c(s, (0, \ldots, 0, 1))\big).$$

More generally, the mapping Φ can be computed directly from c by evaluating c at any set of vectors whose span includes $\mathcal{W}$. We will assume that we have access to Φ and c interchangeably. Note that previous work represented the problem directly as a geometric problem in $\mathbb{R}^n$, but in our case we hope that making the mapping Φ explicit clarifies the algorithm.

An *α-approximation algorithm* A ($\alpha \geq 1$) for such a problem takes as input any vector $w \in \mathcal{W}$ and outputs $A(w) \in \mathcal{S}$ such that $c(A(w), w) \leq \alpha \min_{s\in\mathcal{S}} c(s, w)$. To ensure that the min is well defined, we also assume $\Phi(\mathcal{S}) = \{\Phi(s) \mid s \in \mathcal{S}\}$ is compact.

Define a *projection oracle* $\Pi_J : \mathbb{R}^n \to J$, where $\Pi_J(x) = \mathrm{argmin}_{z\in J} \|x - z\|$ is the unique projection of x to the closest point z in the set J.

Define $\mathcal{W}_+ = \{aw | a \geq 0, w \in \mathcal{W}\} \subseteq \mathbb{R}^n$. Note that $\mathcal{W}_+$ is convex, which follows from the convexity of $\mathcal{W}$. We assume that we have an exact projection oracle $\Pi_{\mathcal{W}_+}$. This is generally straightforward to compute. In many cases, $\mathcal{W} = [0,1]^n$, in which case $\mathcal{W}_+$ is the positive orthant and $\Pi_{\mathcal{W}_+}(w)[i]$ is simply $\max(w[i], 0)$, where $w[i]$ denotes the ith component of vector w. More generally, given a membership oracle to $\mathcal{W}$ (and a point $w_0 \in \mathcal{W}$ and appropriate bounds on the radii of contained and containing balls), one can approximate the projection to within any desired accuracy $\epsilon > 0$ in time $\text{poly}(n, \log(1/\epsilon))$.

We also assume, for convenience, that $A : \mathcal{W}_+ \to \mathcal{S}$ because we know that $A(w)$ can be chosen to be equal to $A(aw)$ for any $a > 0$, and finding a such that $aw \in \mathcal{W}$ is a one-dimensional problem. (Again, given a membership oracle to $\mathcal{W}$ one can find $v \in \mathcal{W}$ which is within ϵ of being a scaled version of w using time $\text{poly}(n, 1/\epsilon)$.) However, the restriction on the approximation algorithm's domain is important because many natural approximation algorithms apply only to restricted domains such as nonnegative weight vectors.

In an *online linear optimization* problem, there is a sequence $w_1, w_2, \ldots \in \mathcal{W}$ of weight vectors. Due to the linearity of the problem, an *offline optimum* can be computed using an exact optimizer, that is, $\min_{s \in \mathcal{S}} \frac{1}{T} \sum_{t=1}^{T} \Phi(s) \cdot w_t = \min_{s \in \mathcal{S}} \Phi(s) \cdot \left(\frac{1}{T} \sum_{t=1}^{T} w_t\right)$ gives the average cost of the best single decision if one had to use a single decision during all time periods $t = 1, 2, \ldots, T$. Similarly, an α-approximation algorithm, when applied to $\frac{1}{T} \sum_{t=1}^{T} w_t$, gives a decision whose average cost is not more than a factor α larger than that of the offline optimum.

DEFINITION 2.2. *In a* full-information online linear optimization problem*, there is an unknown sequence of weight vectors $w_1, w_2, \ldots \in \mathcal{W}$ (possibly chosen by an adversary). On each period, the decision maker chooses a decision $s_t \in \mathcal{S}$ based on $s_1, w_1, s_2, w_2, \ldots, s_{t-1}, w_{t-1}$. Then w_t is revealed and the decision maker incurs cost $c(s_t, w_t)$.*

Finally, we define the bandit version of the problem, in which the algorithm finds out only the cost of its decision, $c(s_t, w_t)$, but *not* w_t itself.

DEFINITION 2.3. *In a* bandit online linear optimization problem*, there is an unknown sequence of weight vectors $w_1, w_2, \ldots \in \mathcal{W}$ (possibly chosen by an adversary). On each period, the decision maker chooses a decision $s_t \in \mathcal{S}$ based only upon $s_1, c(w_1, s_1), \ldots, s_{t-1}, c(w_{t-1}, s_{t-1})$. Then only the cost $c(s_t, w_t)$ is revealed.*

The performance of an online algorithm is measured by comparing its cost on a sequence of weight vectors with the cost of the best static decision for that sequence.

DEFINITION 2.4. *The* α-regret *of an algorithm that selects decisions $s_1, \ldots, s_T \in S$ is defined to be*

$$\alpha\text{-}regret(s_1, w_1, \ldots, s_T, w_T) = \frac{1}{T} \sum_{t=1}^{T} c(s_t, w_t) - \alpha \min_{s \in \mathcal{S}} \frac{1}{T} \sum_{t=1}^{T} c(s, w_t).$$

The term regret *by itself refers to* 1*-regret.*

For $x, y \in \mathbb{R}^n$ and $\mathcal{W} \subseteq \mathbb{R}^n$, we say x *dominates* y if $x \cdot w \leq y \cdot w$ for all $w \in \mathcal{W}$ (equivalently, for all $w \in \mathcal{W}_+$).[4]

Define $K \subseteq \mathbb{R}^n$ to be the convex hull of $\Phi(\mathcal{S})$,

$$K = \left\{ \sum_{i=1}^{n+1} \lambda_i \Phi(s_i) \,\middle|\, s_i \in \mathcal{S}, \lambda_i \geq 0, \sum_i \lambda_i = 1 \right\}.$$

[4]Note that this definition differs from the standard definition in $\mathbb{R}^n$ where x dominates y if $x[i] \geq y[i]$ for all i but resembles the game-theoretic notion of dominant strategies.

Note that $\min_{x \in K} x \cdot w = \min_{s \in \mathcal{S}} c(s, w)$ for all $w \in \mathcal{W}$. The cost of any point in K can be achieved by choosing a randomized combination of decisions $s \in \mathcal{S}$. However, we must find such a combination of decisions and compute projections in our setting, where our only access to $\mathcal{S}$ is via an approximation oracle.

3. Full-information algorithm. We now present our algorithm for the full-information setting. Define $z_t = x_t - \eta w_t$. Intuitively, one might like to play z_t on period $t+1$ because z_t has less cost than x_t against w_t. Unfortunately, z_t may not be feasible. In the Greedy Projection algorithm of Zinkevich, the decision played on period $t+1$ is the projection of z_t into the feasible set. Our basic approach is to implement an approximate projection algorithm and play the approximate projection of z_t on step $(t+1)$.

There are a number of technical challenges to this approach. First, we only have access to an α-approximation oracle with which to implement this. Due to the multiplicative nature of this approximation, we proceed by attempting to project into the set αK, where $\alpha K = \{\alpha x | x \in K\}$. Second, even if we could do this perfectly (which is not possible), this would still not result in a feasible decision. We then must find a way to play a feasible decision.

We can intuitively view our algorithm as follows. The algorithm keeps track of a parameter x_t, which we can think of as the attempt to project z_{t-1} into αK (though this is not done exactly, as x_t is not even in αK). We show that if the algorithm actually were allowed to play x_t, then it would have low α-regret. Our algorithm uses this x_t to find a randomized feasible decision s_t. We show that the expected cost of this random feasible decision s_t is no larger than that of the infeasible x_t.

Our algorithm for the full-information setting is based on the approximate projection routine defined in Figure 3.3.

Algorithm 3.1. *The algorithm is given a learning parameter η. On period 1, we choose an arbitrary s_1 (which could be selected by running the approximation oracle on any input) and let $x_1 = \Phi(s_1)$. On period t, we play s_t and let*

$$(x_{t+1}, s_{t+1}) = \text{Approx-Proj}(x_t - \eta w_t, s_t, x_t).$$

It may be helpful to the reader to note that the sequence x_t is deterministically determined (if the approximation oracle is deterministic) by the sequence of weights $w_1, \ldots, w_{t-1}$, while s_t is necessarily randomized.

In section 3.1, we show that if we had a particular kind of approximate projection algorithm, then the x_t values produced by that algorithm would have (hypothetical) low α-regret. In section 3.2, we show how to extend the domain of any approximation algorithm, which allows us to construct such an approximate projection algorithm: the Approx-Proj algorithm used in Algorithm 3.1. We also show that the cost of the (infeasible) decision x_t it produces can only be larger than the expected cost incurred by the feasible decision s_t it also generates. This will allow us to prove our main theorem in the full-information setting.

Theorem 3.2. *Consider an n-dimensional online linear optimization problem with feasible set $\mathcal{S}$ and mapping $\Phi : \mathcal{S} \to \mathbb{R}^n$. Let A be an α-approximation algorithm and take $R, W \geq 0$ such that $\|\Phi(A(w))\| \leq R$ and $\|w\| \leq W$ for all $w \in \mathcal{W}$.*

For any fixed $w_1, w_2, \ldots, w_T \in \mathcal{W}$ and any $T \geq 1$, with learning parameter $\eta = \frac{(\alpha+1)R}{W\sqrt{T}}$, approximate projection tolerance parameter $\delta = \frac{(\alpha+1)R^2}{T}$, and learning rate

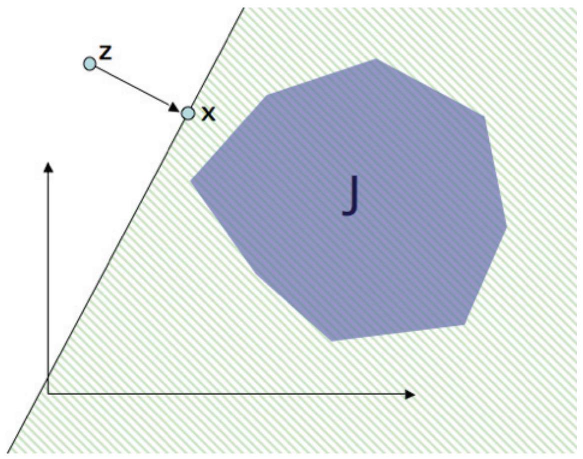

FIG. 3.1. *An approximate projection oracle, for convex set $J \subseteq \mathbb{R}^n$ and $\delta = 0$, returns a point $\Pi_J^0(z) \in \mathbb{R}^n$ that is closer to any point $y \in J$ than z is, that is, for all $y \in J$ $\|\Pi_J^0(z) - y\| \leq \|z - y\|$.*

parameter $\lambda = \frac{(\alpha+1)}{4(\alpha+2)^2 T}$, Algorithm 3.1 *achieves expected α-regret at most*

$$\mathrm{E}\left[\frac{1}{T}\sum_{t=1}^{T} c(s_t, w_t)\right] - \alpha \min_{s \in \mathcal{S}} \frac{1}{T}\sum_{t=1}^{T} c(s, w_t) \leq \frac{(\alpha+2)RW}{\sqrt{T}}.$$

On each period, the algorithm makes at most $4(\alpha+2)^2 T$ calls to A and Φ.

We present the proof of Theorem 3.2 in section 3.3. To get Result 1.1 in the introduction, we note that it is possible to get a priori bounds on W and R by a simple change of basis so that $RW = O(n)$. It is possible to do this from the set $\mathcal{W}$ alone. In particular, one can compute a 2-barycentric spanner (BS) $e_1, \ldots, e_n$ for $\mathcal{W}$ [2] and perform a change of basis so that $\Phi(e_1), \ldots, \Phi(e_n)$ is the standard basis (as we describe in more detail in section 4). By the definition of a 2-BS, this implies that $\mathcal{W} \subseteq [-2, 2]^n$, and hence $W = 2\sqrt{n}$ is a satisfactory upper bound. Since we have assumed that all costs are in $[0, 1]$ and the standard basis is in $\mathcal{W}$, this implies that $\Phi(S) \subseteq [0, 1]^n$, and hence $R = \sqrt{n}$ is also a valid upper bound. The guarantees with respect to every window of T consecutive periods hold because our algorithm's guarantees hold starting at arbitrary (s_t, x_t) such that $\mathrm{E}[\Phi(s_t)]$ dominates x_t (recall that s_t is necessarily randomized).

3.1. Approximate projection. We first define the notion of approximate projection. Because we only have access to an α-approximate oracle, given $z \in \mathbb{R}^n$, we cannot find the closest point to z in K or even in $\alpha K = \{\alpha x | x \in K\}$.

Note that for a closed convex set $J \subseteq \mathbb{R}^n$, if $\Pi_J(z) = x$, then

$$(x - z) \cdot x \leq \min_{y \in J} (x - z) \cdot y.$$

This is essentially the separating hyperplane theorem (where $x - z$ is the normal vector to the separating hyperplane). Also note that $\Pi_J(x) = x$ if $x \in J$.

Our approximate projection property, illustrated in Figure 3.1, relaxes the above condition. Due to the computational issues associated with optimizing over K even with access to an *exact* optimization oracle ($\alpha = 1$),[5] our projections will be parametrized by an additional δ. Define the set of δ-approximate projections to be, for $\delta \geq 0$ and any $z \in \mathbb{R}^n$,

$$\Pi_J^\delta(z) = \Big\{x \in \mathbb{R}^n \mid (x - z) \cdot x \leq \min_{y \in J}(x - z) \cdot y + \delta\Big\}.$$

It is important to note that we have not required an approximate projection to be in J. However, note that in the case where the projection is in J, and $\delta = 0$, it is exactly the projection, that is, $\Pi_J^\delta(z) \cap J = \{\Pi_J(z)\}$. While we refer to it as an approximate projection, it is also clearly related to a separation oracle. From a hyperplane separating z from J, one can take the closest point on that hyperplane to z as an approximate projection. The difficulty is in finding a feasible such point.

We now bound the α-regret of the hypothetical algorithm which projects with $\Pi_{\alpha K}^\delta$. The proof is essentially a straightforward extension of Zinkevich's proof [16]. This lemma shows that indeed this hypothetical algorithm has a graceful degradation in quality.

LEMMA 3.3. *Let $K \subseteq \mathbb{R}^n$ be a convex set such that for all $x \in K$, $\|x\| \leq R$. Let $w_1, \ldots, w_T \in \mathbb{R}^n$ be an arbitrary sequence. Then, for any initial point $x_1 \in K$ and any sequence $x_1, x_2, \ldots, x_T$ such that $x_{t+1} \in \Pi_{\alpha K}^\delta(x_t - \eta w_t)$,*

$$\frac{1}{T}\sum_{t=1}^{T} x_t \cdot w_t - \alpha \min_{x \in K} \frac{1}{T}\sum_{t=1}^{T} x \cdot w_t \leq \frac{(\alpha+1)^2 R^2}{2\eta T} + \frac{\eta}{2T}\sum_{t=1}^{T} w_t^2 + \frac{\delta}{\eta}.$$

Proof. Let $x^* = \alpha \operatorname{argmin}_{x \in K} \sum_{t=1}^T x \cdot w_t$, so $x^* \in \alpha K$. We will bound our performance with respect to x^*. Define the sequence x'_t by $x'_1 = x_1$ and $x'_{t+1} = x_t - \eta w_t$, so that $x_t \in \Pi_{\alpha K}^\delta(x'_t)$. We first claim that $\|x_t - x^*\|^2 \leq \|x'_t - x^*\|^2 + 2\delta$; that is, our attempt at setting x_t to be an approximate projection of x_t onto αK does not increase the distance to x^* significantly:

$$\begin{aligned}(x'_t - x^*)^2 &= \big((x'_t - x_t) + (x_t - x^*)\big)^2 \\ &= (x'_t - x_t)^2 + (x_t - x^*)^2 + 2(x'_t - x_t)\cdot(x_t - x^*) \\ &\geq 0 + (x_t - x^*)^2 - 2\delta.\end{aligned}$$

The last line follows from the definition of approximate projection and the fact that $x^* \in \alpha K$.

Hence, for any $t \geq 1$, because $x'_{t+1} = x_t - \eta w_t$ we have

$$\begin{aligned}(x_{t+1} - x^*)^2 &\leq (x_t - \eta w_t - x^*)^2 + 2\delta \\ &= (x_t - x^*)^2 + \eta^2 w_t^2 - 2\eta w_t \cdot (x_t - x^*) + 2\delta\end{aligned}$$

and thus

$$w_t \cdot (x_t - x^*) \leq \frac{(x_t - x^*)^2 - (x_{t+1} - x^*)^2 + \eta^2 w_t^2 + 2\delta}{2\eta}.$$

Using a telescoping sum of the above and the fact that

$$(x_1 - x^*)^2 \leq (\|x_1\| + \|x^*\|)^2 \leq (\alpha+1)^2 R^2,$$

[5] We are not assuming that K is defined by a finite number of hyperplanes—it can be quite round.

we get

$$\sum_{t=1}^{T} x_t \cdot w_t - \alpha \min_{x \in K} \sum_{t=1}^{T} x \cdot w_t \leq \frac{(\alpha+1)^2 R^2}{2\eta} + \frac{\eta}{2} \sum_{t=1}^{T} w_t^2 + T\frac{\delta}{\eta}$$

as desired. ☐

Note that if we set $\eta = 1/\sqrt{T}$, the sum of the first two terms of this bound would be $O(1/\sqrt{T})$. However, the last term, $\frac{\delta}{\eta}$, would be $O(\delta\sqrt{T})$. Hence, we need to achieve an approximation quality of $\delta = O(1/T)$ in order for the α-regret of our (infeasible) x_t values to be $O(1/\sqrt{T})$.

3.2. Constructing the algorithm. One simple method to (approximately) find a projection of z into a convex set J, given an exact optimization oracle for J, is as follows. Start with a point in $x \in J$. Then choose the search direction $v = x - z$, and find a minimal point $x' \in J$ in the direction of v, that is, $x' \in J$ such that $x' \cdot v \leq \min_{y \in J} y \cdot v$ (or, equivalently, such that $(x' - z) \cdot v \leq \min_{y \in J}(y - z) \cdot v$). It can be seen that if x is not minimal in the direction of v, then there must be a point on the segment joining x' and z that is closer to z than x was. Then repeat this procedure starting at x'. In the case where $z \in J$, this will still be useful in representing z nearly as a combination of points output by the minimization algorithm.[6]

Note that in our case if $v \in \mathcal{W}_+$, then our approximation oracle is able to find a feasible $s \in \mathcal{S}$ such that

$$\Phi(s) \cdot v \leq \alpha \min_{s' \in \mathcal{S}} \Phi(s') \cdot v = \min_{x \in \alpha K} x \cdot v.$$

Loosely speaking, our oracle is able to perform minimization with respect to the set $J = \alpha K$ (or better). This is essentially how our algorithm will use the approximation oracle. However, as mentioned before, many approximation algorithms can only handle nonnegative weight vectors or weight vectors from some other limited domain. Hence, we must extend the domain of the oracle when $v \notin \mathcal{W}_+$.

Extending the domain. We would like to find a feasible $s \in \mathcal{S}$ that satisfies the search condition $\Phi(s) \cdot v \leq \alpha \min_{s' \in \mathcal{S}} \Phi(s') \cdot v$ for a general $v \in \mathbb{R}^n$, but this is not possible given only an α-approximation oracle that runs on only a subset of $\mathbb{R}^n$. Instead, we attempt to find a (potentially infeasible) $x \in \mathbb{R}^n$ which does satisfy this search condition, and we also attempt to find an $s \in \mathcal{S}$ which dominates x, meaning that for all $w \in \mathcal{W}$, $c(s, w) \leq x \cdot w$. More precisely, we will construct the following oracle.

DEFINITION 3.4. *An* extended approximation oracle $B : \mathbb{R}^n \to \mathcal{S} \times \mathbb{R}^n$ *is a function such that, for all* $v \in \mathbb{R}^n$*, if* $B(v) = (s, x)$*, then* $x \cdot v \leq \alpha \min_{s' \in \mathcal{S}} \Phi(s') \cdot v$ *and* $\Phi(s)$ *dominates* x*.*

Figure 3.2 depicts an extended approximation oracle. The following lemma demonstrates that one can construct an extended approximation oracle from an approximation oracle.

LEMMA 3.5. *Let* $A : \mathcal{W}_+ \to \mathcal{S}$ *be an* α*-approximation oracle and suppose* $\|\Phi(s')\| \leq R$ *for all* $s' \in \mathcal{S}$*. Then the following is an extended approximation*

[6]Note that representing a given feasible point as a convex combination of feasible points is similar to *randomized metarounding* [5]. It would be interesting to extend their approach, based on the ellipsoid algorithm, to our problem and potentially achieve a more efficient algorithm. Related but simpler issues arise in [6].

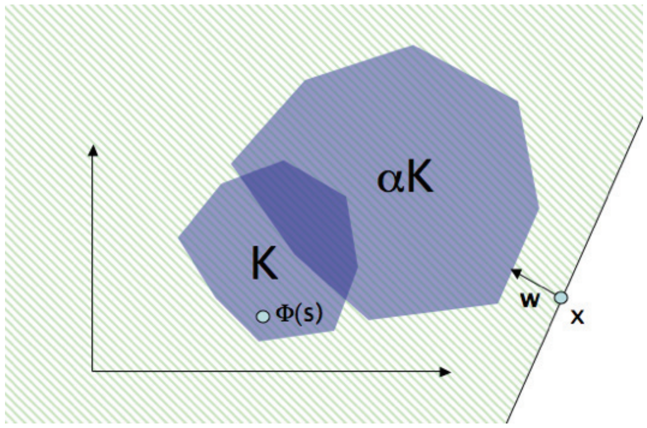

FIG. 3.2. *An approximation algorithm run on vector $w \in \mathcal{W}$ always returns a point $s \in \mathcal{S}$ such that the set αK is contained in the half-space tangent to $\Phi(s)$ whose normal direction is w. An extended approximation algorithm, as illustrated here, takes any $w \in \mathbb{R}^n$ as input and returns a point $x \in \mathbb{R}^n$ such that αK is contained in the half-space tangent to x with normal vector w. In addition, it returns an $s \in \mathcal{S}$ such that $\Phi(s)$ dominates x.*

oracle: If $v \in \mathcal{W}_+$, then $B(v) = (A(v), \Phi(A(v)))$; else $B(v)$ is

$$\left(A(\Pi_{\mathcal{W}_+}(v)), \Phi(A(\Pi_{\mathcal{W}_+}(v))) + R(\alpha+1)\frac{\Pi_{\mathcal{W}_+}(v) - v}{||\Pi_{\mathcal{W}_+}(v) - v||} \right).$$

Proof. For the case where $v \in \mathcal{W}_+$, by definition, $B(v) = (A(v), \Phi(A(v)))$ suffices. Hence, assume $v \notin \mathcal{W}_+$. Let $w = \Pi_{\mathcal{W}_+}(v)$, $s = A(w)$, and $x = \Phi(s) + (\alpha+1)R\frac{w-v}{||w-v||}$. Then we must show that (a) $x \cdot v \leq \alpha \min_{s' \in \mathcal{S}} \Phi(s') \cdot v$ and (b) $\Phi(s)$ dominates x.

We have assumed that A is an α-approximation oracle with domain $\mathcal{W}_+$, and therefore it can accept input w. By the definition of α-approximation, we have $w \cdot \Phi(s) \leq \alpha w \cdot \Phi(s')$ for all $s' \in \mathcal{S}$. By the bound R, we also have that $-\alpha\|v-w\|R \leq \alpha(v-w)\cdot\Phi(s')$ for all $s' \in \mathcal{S}$. Adding these two gives, for all $s' \in \mathcal{S}$,

$$\begin{aligned}
\alpha v \cdot \Phi(s') &\geq w \cdot \Phi(s) - \alpha\|v-w\|R \\
&= v \cdot x + (w-v)\cdot\Phi(s) - (\alpha+1)R\frac{(w-v)}{\|w-v\|}\cdot v - \alpha\|v-w\|R \\
&\geq v \cdot x - \|w-v\|R - (\alpha+1)R\frac{(w-v)}{\|w-v\|}\cdot(v-w) - \alpha\|v-w\|R \\
&= v \cdot x.
\end{aligned}$$

This is what we need for part (a) of the lemma. The second-to-last line follows from the fact that $(v-w)\cdot w = 0$. To see this, note that since w is the projection of v onto $\mathcal{W}_+$, we have $(v-w)\cdot(w'-w) \leq 0$ for any $w' \in \mathcal{W}_+$. Since $0 \in \mathcal{W}_+$, this implies that $(v-w)\cdot(-w) \leq 0$. Since $2w \in \mathcal{W}_+$, this implies that $(v-w)\cdot w \leq 0$, and hence $(v-w)\cdot w = 0$.

This also means that $(v-w)\cdot(w'-w) = (v-w)\cdot w' \leq 0$ for all $w' \in \mathcal{W}_+$, which directly implies (b), that is, $(x - \Phi(s))\cdot w' \geq 0$ for all $w' \in \mathcal{W}$. ☐

Note that the magnitude of the output x is at most $\|\Phi(s)\| + (\alpha+1)R \leq (\alpha+2)R$; this bound will be useful for bounding the runtime of our algorithm.

```
Input: $x, z \in \mathbb{R}^n$, $s \in \mathcal{S}$, and an $\alpha$-approximation algorithm $A$ (and parameters $\delta > 0$,
$\lambda \in [0,1]$).
Output: $(x', s') \in \Pi^\delta_{\alpha K} \times \mathcal{S}$
Define $B$ to be the extended approximation oracle obtained from $A$ using Lemma 3.5.
APPROX-PROJ$(z, s, x)$
1  Let $(t, y) := B(x - z)$
2  if $x \cdot (x - z) \le \delta + y \cdot (x - z)$
3      then return $(x, s)$
4      else $q = \begin{cases} s & \text{with probability } 1 - \lambda \\ t & \text{with probability } \lambda \end{cases}$
5          return APPROX-PROJ$(z, q, \lambda y + (1 - \lambda)x)$
```

FIG. 3.3. *A recursive algorithm for computing approximate projections.*

The approximate projection algorithm. Using this extended approximation oracle, we can define our APPROX-PROJ algorithm, which we present in Figure 3.3. The following lemma shows that the algorithm returns both a valid approximate projection (which could be infeasible) and a random feasible decision that dominates the approximate projection (assuming that Φ of the algorithm's input s dominated the algorithm's input x).

LEMMA 3.6. *Suppose* APPROX-PROJ(z, s, x) *returns* (x', s'). *Then* $x' \in \Pi^\delta_{\alpha K}(z)$. *If s is a random variable such that* $\mathrm{E}[\Phi(s)]$ *dominates x, then* $\mathrm{E}[\Phi(s')]$ *will dominate x'.*

It is straightforward to see that the x returned by APPROX-PROJ satisfies the approximate projection condition. The subtlety is in obtaining a feasible solution with the desired properties. It turns out that t returned by B in line 1 does not suffice, as this t dominates only y, but not necessarily x. However, our randomized scheme does suffice.

Proof of Lemma 3.6. The return condition of APPROX-PROJ states that $x' \cdot (x' - z) \le \delta + y \cdot (x' - z)$. Using the definition of an extended approximation oracle, we then get

$$\begin{aligned} x' \cdot (x' - z) &\le \delta + \alpha \min_{s' \in \mathcal{S}} \Phi(s') \cdot (x' - z) \\ &\le \delta + \min_{y' \in \alpha K} y' \cdot (x' - z) \end{aligned}$$

as desired.

The proof of the second property proceeds by induction on the number of recursive calls made by APPROX-PROJ. The base case holds trivially. Now suppose the inductive hypothesis holds ($\mathrm{E}[\Phi(s)]$ dominates x). We will show that if $(t, y) = B(x - z)$, the resulting $\mathrm{E}[\lambda\Phi(t) + (1 - \lambda)\Phi(s)]$ dominates $\lambda y + (1 - \lambda)x$.

We observe that

$$\begin{aligned} x' \cdot w &= (\lambda y + (1 - \lambda)x) \cdot w \\ &= \lambda y \cdot w + (1 - \lambda)x \cdot w \\ &\ge \lambda\Phi(t) \cdot w + (1 - \lambda)x \cdot w \\ &\ge \lambda\Phi(t) \cdot w + (1 - \lambda)\mathrm{E}[\Phi(s)] \cdot w \\ &= \mathrm{E}[\lambda\Phi(t) + (1 - \lambda)\Phi(s)] \cdot w \\ &= \mathrm{E}[\Phi(s')] \cdot w. \end{aligned}$$

Thus, if APPROX-PROJ terminates, the desired conditions will hold. ∎

3.3. Analysis. Our next lemma allows us to bound the number of calls Algorithm 3.1 makes to A and Φ on each period.

LEMMA 3.7. *Suppose that $\lambda, \delta > 0$ and the magnitudes of all vectors output by the extended approximation oracle are $\leq \frac{1}{2}\sqrt{\delta/\lambda}$ and $\|x\| \leq \frac{1}{2}\sqrt{\delta/\lambda}$. Then* APPROX-PROJ*$(z, s, x)$ terminates after at most $\frac{\|x-z\|^2}{\delta\lambda}$ iterations.*

Proof. The analysis is reminiscent of that of the perceptron algorithm (see, e.g., Dunagan and Vempala [9]). Let $H = \frac{1}{2}\sqrt{\delta/\lambda}$. To bound the number of recursive calls to APPROX-PROJ, it suffices to show that the nonnegative quantity $\|x - z\|^2$ decreases by at least an additive $\lambda\delta$ on each call and that $\|x\|$ remains below H on successive calls. The latter condition holds because $\|x\|, \|y\| \leq H$, so $\|\lambda y + (1-\lambda)x\| \leq \lambda H + (1-\lambda)H = H$.

Notice that if the procedure does not terminate on a particular call, then

$$(x - y) \cdot (x - z) > \delta.$$

This means that the decrease in $(x - z)^2$ in a single recursive call is

$$\begin{aligned}(x-z)^2 - (\lambda y + (1-\lambda)x - z)^2 &= (x-z)^2 - (\lambda(y-x) + (x-z))^2 \\ &= 2\lambda(x-y)\cdot(x-z) - \lambda^2(y-x)^2 \\ &> 2\lambda\delta - \lambda^2(y-x)^2.\end{aligned}$$

Also, $\|y - x\| \leq 2H$. Combining this with the previous observation gives

$$(x-z)^2 - (\lambda y + (1-\lambda)\,x - z)^2 > 2\lambda\delta - \lambda^2 4H^2 = \lambda\delta.$$

Hence the total number of iterations of APPROX-PROJ on each period is at most $\|x - z\|^2/(\lambda\delta)$. ◻

This lemma gives us a means of choosing λ. We are now ready to prove our main theorem about full-information online optimization.

Proof of Theorem 3.2. Take $\eta = \frac{(\alpha+1)R}{W\sqrt{T}}$, $\delta = \frac{(\alpha+1)R^2}{T}$, and $\lambda = \frac{(\alpha+1)}{4(\alpha+2)^2T}$. Since $x_1 = \Phi(s_1)$, by induction and by Lemma 3.6, we have that $\mathrm{E}[\Phi(s_t)]$ dominates x_t for all t. Hence, it suffices to upper-bound $\sum_{t=1}^{T} x_t \cdot w_t$. By Lemma 3.6, we have that $x_t \in \Pi^{\delta}_{\alpha K}(z_{t-1})$ on each period, so by Lemma 3.3 we get

$$\mathrm{E}[\alpha\text{-regret}] \leq \frac{1}{T}\left(\frac{(\alpha+1)^2R^2}{2\eta} + T\frac{\delta}{\eta} + \frac{\eta}{2}TW^2\right).$$

Applying our chosen values of η and δ, this gives an α-regret bound of

$$\frac{1}{T}((\alpha+1)RW\sqrt{T} + RW\sqrt{T}) = \frac{(\alpha+2)RW}{\sqrt{T}}$$

as desired.

Now, as mentioned, the extended approximation oracle from Lemma 3.5 has the property that it returns vectors of magnitude at most $H = \frac{1}{2}\sqrt{\delta/\lambda} = (\alpha+2)R$. Furthermore, it is easy to see that all vectors x_t have $\|x_t\| \leq H$, by induction on t. Then by Lemma 3.7, the total number of iterations of APPROX-PROJ on period t is at most $(2H\|x - z\|/\delta)^2 \leq (2(\alpha+2)R\eta W/\delta)^2 = 4(\alpha+2)^2T$. ◻

4. Bandit algorithm.

We now describe how to extend Algorithm 3.1 to the partial-information model, where the only feedback we receive is the cost we incur at each period. Flaxman, Kalai, and McMahan [10] also used a gradient descent style algorithm for online optimization in the bandit setting, but the details of their approach differ significantly from ours. The algorithm we describe here requires access to an *exploration basis* $e_1, \ldots, e_n \in \mathcal{S}$, which is simply a set of n decisions such that $\Phi(e_1), \ldots, \Phi(e_n)$ span $\mathbb{R}^n$. (If no such decisions exist, one can reduce the problem to a lower-dimensional problem.) Following previous approaches, we will (probabilistically) try each of these decisions from time to time. As in the work of Dani and Hayes [7], we will assume that $\Phi(e_i)$ is the standard ith basis vector, that is, $e_i[i] = 1$ and $e_i[j] = 0$ for $j \neq i$. This assumption makes the algorithm cleaner to present, and is without loss of generality, because we can always use $\Phi(e_i)$ as our basis for representing $\mathbb{R}^n$.

DEFINITION 4.1. *A set $\{x_1, x_2, \ldots, x_m\} \subseteq S$ is a β-BS for $S \subset \mathbb{R}^n$ if, for every $x \in S$, x can be written as $x = \beta_1 x_1 + \cdots + \beta_m x_m$ for some $\beta_1, \ldots, \beta_m \in [-\beta, \beta]$.*

Note that we need only construct a BS once for any problem, and then we can reuse it for all future instances of the problem.

Awerbuch and Kleinberg [2] proved that every compact S has a 1-BS of size n, and, moreover, they gave an algorithm for finding a size-n $(1+\epsilon)$-BS using poly$(n, \log(1/\epsilon))$ calls to an exact minimization oracle $M : \mathbb{R}^n \to \mathcal{S}$, where $M(v) \in \operatorname{argmin}_{s\in\mathcal{S}} \Phi(s) \cdot v$. Unfortunately, as we show in section 4.1, one cannot find such a BS using a minimizer (exact or approximate) whose domain is not all of $\mathbb{R}^n$. Moreover, we show that one cannot guarantee low regret for the bandit problem using just a black-box optimization algorithm $A : \mathcal{W}_+ \to \mathcal{S}$.

Hence, we assume that we are given a β-BS for the problem at hand as part of the input. We feel that this is a reasonable assumption. For example, note that it is easy to find such a basis for TSP and set cover with $\beta =$poly(n): In the case of set cover, one can take the n covers consisting of all sets but one.[7] In the case of TSP, we can start with any tour σ that visits all the edges at least once and consider σ_e for each edge e which is the same as σ but traverses e an additional two times.

We present the algorithm for the bandit setting in Figure 4.1. We remark that our approach is essentially the same as previous approaches and can be used as a generic conversion from a black-box full-information online algorithm to a bandit algorithm. Previous approaches also worked in this manner, but the analysis depended on the specific bounds of the black-box algorithm in a way that, unfortunately, we cannot simply reference.

THEOREM 4.2. *For $\alpha, \beta \geq 1$, integer $T \geq 0$, and any $w_1, \ldots, w_T$, given an α-approximation oracle and a β-BS, the algorithm in Figure 4.1 with $\eta = \frac{(\alpha+1)R}{D\sqrt{T}}$, $\delta = \eta n T^{-1/3}$, and $\gamma = (4\alpha\beta)^{2/3} n T^{-1/3}$ achieves an expected α-regret bound in the bandit setting of*

$$\mathrm{E}[\alpha\text{-}regret] \leq 7n(\alpha\beta)^{2/3} T^{-1/3}.$$

The conversion from full-information to bandit is similar to other conversions [2, 14, 7]. Note that in the description of the algorithm, s_t is what is played at step t. Also note that $\hat{x}_{t+1}$ may be viewed as an approximate projection of $\hat{x}_t$ when it is

[7]If any of these is not a cover, that set must be mandatory in any cover and we can simplify the problem. If this set of covers is not linearly independent, then we can reduce the dimensionality of the problem and use the fact that if T is a (possibly linearly dependent) β-BS for S and R is a γ-BS for T, then R is a $(\gamma\beta|T|)$-BS for S.

```
Given $\delta, \eta, \gamma > 0$ and an initial point $\hat{s}_1$ as input, set $\hat{x}_1 = \Phi(\hat{s}_1)$. Perform a change
of basis so that $\Phi(e_1), \ldots, \Phi(e_n)$ is the standard basis.
for $t = 1, 2, \ldots$:
    With probability $\gamma$,        $\triangleright$ exploration step
            Choose $i \in \{1, \ldots, n\}$ uniformly at random.
            $s_t := e_i; x_t := \Phi(e_i)$.
            Play$(s_t)$.
            Observe $\ell_t = c(s_t, w_t)$.
            $\hat{w}_t := (n\ell_t/\gamma)\Phi(e_i)$.
            $(\hat{x}_{t+1}, \hat{s}_{t+1}) :=$ APPROX-PROJ$(\hat{x}_t - \eta\hat{w}_t, \hat{s}_t, \hat{x}_t)$.
    else, with probability $1 - \gamma$,        $\triangleright$ exploitation step
            $s_t := \hat{s}_t; x_t := \hat{x}_t$.
            Play$(s_t)$.
            Observe $\ell_t = c(s_t, w_t)$.
            $\hat{w}_t := 0$.
            $(\hat{x}_{t+1}, \hat{s}_{t+1}) := (\hat{x}_t, \hat{s}_t)$.
```

FIG. 4.1. *Algorithm for the bandit setting.*

generated in exploitation steps as well as in exploration steps, since $\hat{x}_t \in \Pi^{\delta}_{\alpha J}(\hat{x}_t - \eta\hat{w}_t)$ for $\hat{w}_t = 0$. We first prove the following lemma.

LEMMA 4.3. *Let $J \subseteq \mathbb{R}^n$ be a convex set such that for all $\hat{x} \in J$, $\|\hat{x}\| \le R$. Let $w_1, \ldots, w_T \in \mathbb{R}^n$ be an arbitrary sequence and let $\hat{w}_1, \ldots, \hat{w}_T$ be a sequence of random variables such that $\mathrm{E}[\hat{w}_t|\hat{x}_1, \hat{w}_1, \ldots, \hat{x}_{t-1}, \hat{w}_{t-1}, \hat{x}_t] = w_t$ and $\mathrm{E}[\hat{w}_t^2] \le D^2$. Then, for any initial point $\hat{x}_1 \in J$ and any sequence $\hat{x}_1, \hat{x}_2, \ldots$ such that $\hat{x}_{t+1} \in \Pi^{\delta}_{\alpha J}(\hat{x}_t - \eta\hat{w}_t)$,*

$$\mathrm{E}\left[\sum_{t=1}^{T} \hat{x}_t \cdot w_t\right] - \alpha \min_{x \in J} \sum_{t=1}^{T} x \cdot w_t \le \frac{(\alpha+1)^2 R^2}{2\eta} + T\frac{\delta}{\eta} + \frac{\eta}{2} D^2 T + 2\alpha R D\sqrt{T}.$$

Proof. By Lemma 3.3, we have that

$$\sum_{t=1}^{T} \hat{x}_t \cdot \hat{w}_t - \alpha \min_{x \in J} \sum_{t=1}^{T} x \cdot \hat{w}_t \le \frac{(\alpha+1)^2 R^2}{2\eta} + T\frac{\delta}{\eta} + \frac{\eta}{2} \sum_{t=1}^{T} \hat{w}_t^2.$$

Taking expectations of both sides gives

$$\sum_{t=1}^{T} \hat{x}_t \cdot w_t - \alpha \mathrm{E}\left[\min_{x \in J} \sum_{t=1}^{T} x \cdot \hat{w}_t\right] \le \frac{(\alpha+1)^2 R^2}{2\eta} + T\frac{\delta}{\eta} + \frac{\eta}{2} D^2 T.$$

It thus suffices to show that

$$\mathrm{E}\left[\min_{x \in J} \sum_{t=1}^{T} x \cdot \hat{w}_t\right] \ge \min_{x \in J} \sum_{t=1}^{T} x \cdot w_t - 2RD\sqrt{T}. \tag{4.1}$$

Now, for any $x \in J$,

$$\begin{aligned} \left|\sum_{t=1}^{T} x \cdot (\hat{w}_t - w_t)\right| &\le |x| \left|\sum_{t=1}^{T} \hat{w}_t - w_t\right| \\ &\le R \left|\sum_{t=1}^{T} \hat{w}_t - w_t\right|. \end{aligned} \tag{4.2}$$

This gives us a means of upper-bounding the difference between the minima. Namely,

$$\mathrm{E}\left[\ \left|\sum_{t=1}^{T}\hat{w}_t - w_t\right|\ \right]^2 \leq \mathrm{E}\left[\left(\sum_{t=1}^{T}\hat{w}_t - w_t\right)^2\right]$$

$$= \sum_{t=1}^{T}\mathrm{E}\left[(\hat{w}_t - w_t)^2\right]. \tag{4.3}$$

The last equality follows from the fact that

$$\mathrm{E}[(\hat{w}_{t_1} - w_{t_1})(\hat{w}_{t_2} - w_{t_2})] = 0$$

for $t_1 < t_2$, which follows from the martingale-like assumption that $\mathrm{E}[\hat{w}_{t_2} - w_{t_2}| \hat{w}_{t_1}, w_{t_1}] = 0$. Finally,

$$\begin{aligned}\mathrm{E}[(\hat{w}_t - w_t)^2] &\leq \mathrm{E}[\hat{w}_t^2 + 2\|\hat{w}_t\|\|w_t\| + w_t^2]\\ &\leq D^2 + 2D^2 + D^2\\ &= 4D^2.\end{aligned}$$

In the above we have used the facts that $\mathrm{E}[|\hat{w}_t|]^2 \leq \mathrm{E}[\hat{w}_t^2] \leq D^2$ and $\|w_t\|^2 = \mathrm{E}[\hat{w}_t]^2 \leq \mathrm{E}[\hat{w}_t^2] \leq D^2$. Hence, we have that the quantity in (4.3) is upper-bounded by $4TD^2$, which, together with (4.2), establishes (4.1). $\square$

Proof of Theorem 4.2. We remark that the parameter γ in the statement of the theorem may be larger than 1, but in this case the regret bound is greater than 1 and hence holds for any algorithm.

Note that in the conversion algorithm the expected value of $\hat{w}_t$ is w_t, and this is true conditioned on all previous information as well as $\hat{x}_t$. Since Lemma 3.6 implies $\hat{x}_{t+1} \in \Pi^{\delta}_{\alpha J}(\hat{x}_t - \eta\hat{w}_t)$, we can apply Lemma 4.3 to the sequence $\hat{x}_t$. This gives

$$\sum_{t=1}^{T}\mathrm{E}[\hat{x}_t \cdot w_t] - \alpha\min_{x\in J}\sum_{t=1}^{T} x\cdot w_t \leq \frac{(\alpha+1)^2R^2}{2\eta} + T\frac{\delta}{\eta} + \frac{\eta}{2}D^2T + 2\alpha RD\sqrt{T}.$$

To apply the lemma, we use the bound $D = n\gamma^{-1/2}$. This holds because $\ell_t \in [0,1]$, so $\mathrm{E}[\hat{w}_t^2] \leq \gamma(n\ell_t/\gamma)^2 + (1-\gamma)0 \leq n^2/\gamma$. Also, we use the bound of $R = \beta\sqrt{n}$. Hence we choose $\eta = \frac{(\alpha+1)R}{D\sqrt{T}}$ and $\delta = \eta nT^{-1/3}$, which simplifies the above equation to

$$\begin{aligned}\sum_{t=1}^{T}\mathrm{E}[\hat{x}_t \cdot w_t] - \alpha\min_{x\in J}\sum_{t=1}^{T} x\cdot w_t &\leq (\alpha+1)RD\sqrt{T} + nT^{2/3} + 2\alpha RD\sqrt{T}\\ &\leq 4\alpha RD\sqrt{T} + nT^{2/3}.\end{aligned}$$

Substituting the values of D and R gives an upper bound of $4\alpha\beta n^{3/2}\gamma^{-1/2}\sqrt{T} + T\frac{\delta}{\eta}$.

Next, as in the analysis of the full-information algorithm, $\mathrm{E}[\Phi(\hat{s}_t)]$ dominates $\mathrm{E}[\hat{x}_t]$ by Lemma 3.6. Thus,

$$\sum_{t=1}^{T}\mathrm{E}[c(\hat{s}_t, \cdot w_t)] - \alpha\min_{x\in J}\sum_{t=1}^{T} x\cdot w_t \leq 4\alpha\beta n^{3/2}\gamma^{-1/2}\sqrt{T} + nT^{2/3}.$$

Finally, we have that $\mathrm{E}[c(s_t, w_t)] \leq \mathrm{E}[c(\hat{s}_t, w_t)] + \gamma$ because with probability $1 - \gamma$, $\hat{s}_t = s_t$ and in the remaining case the cost is in $[0, 1]$. Putting these together implies

$$\sum_{t=1}^{T} \mathrm{E}[c(s_t, \cdot w_t)] - \alpha \min_{x \in J} \sum_{t=1}^{T} x \cdot w_t \leq 4\alpha\beta n^{3/2}\gamma^{-1/2}\sqrt{T} + nT^{2/3} + \gamma T.$$

Choosing $\gamma = (4\alpha\beta)^{2/3} nT^{-1/3}$ (note that if this quantity is larger than 1, then the regret bound in the theorem is trivial) gives a bound of $2n(4\alpha\beta T)^{2/3} + nT^{2/3} \leq 7n(\alpha\beta T)^{2/3}$ as in the theorem. ▯

4.1. Difficulty of the black-box reduction. We now point out that it is impossible to solve the bandit problem with general algorithms (approximation or exact) without an exploration basis (that is, if our only access to $\mathcal{S}$ is through a black-box optimization oracle). The counterexample is randomized. We will take

$$\mathcal{W} = \{w \in \mathbb{R}^n \mid w[1] \in [0, 1] \text{ and } \|w\|^2 \leq 2(w[1])^2\}.$$

The set $\mathcal{S}$ will consist of two points: $s = (1/2, 0, \ldots, 0)$ as well as a second point $s' = (1, 0, \ldots, 0) - u$, where $\|u\| = 1$ and $u[1] = 0$. The mapping Φ is the identity mapping. The cost sequence will be constant $w_t = (1, 0, \ldots, 0) + u$. Hence $c(s, w_t) = 1/2$ while $c(s', w_t) = 0$. Now, suppose we, as algorithm designers, know that this is the setup but u is chosen uniformly at random from the set of unit vectors with $u[1] = 0$.

Observation 4.4. *For any bandit algorithm that makes k calls to black-box optimization oracle A, and for any $\alpha \geq 0$, with probability $1 - ke^{-0.1n}$ over u, the algorithm has α-regret $1/2$ on a sequence of arbitrary length.*

Proof. No information is conveyed by the costs returned in the bandit setup of our example—they are always $1/2$ if s' has not been discovered, while the minimal cost is 0. Thus the algorithm must find some $w \in \mathcal{W}$ such that $c(s, w) > c(s', w)$ (whence an exact optimization algorithm must return s'). Without loss of generality, we can scale w so that $w[1] = 1$ and $\|w\| \leq 2$. Hence, we can write $w = (1, 0, 0, \ldots, 0) + v$, where $v[1] = 0$ and $\|v\| \leq 1$. In this case, $w \cdot s = 1/2$, while $w \cdot s' = 1 - u \cdot v$. For u a random unit vector and any fixed $\|v\| \leq 1$, it is known that $\Pr[u \cdot v \geq 1/2]$ is exponentially small in n. A very loose bound can be seen directly, since for a ball of dimension n, this probability is

$$\frac{\int_{1/2}^{1} (\sqrt{(1 - x^2)})^{n-2} dx}{\int_{-1}^{1} (\sqrt{(1 - x^2)})^{n-2} dx} \leq \frac{\int_{1/2}^{1} (3/4)^{\frac{n-2}{2}} dx}{\int_{-1/\sqrt{n}}^{1/\sqrt{n}} (1 - n^{-1})^{\frac{n-2}{2}} dx}$$
$$\leq \frac{\sqrt{ne}}{2} \left(\frac{3}{4}\right)^{\frac{n}{2} - 1},$$

which is $O(e^{-0.1n})$. ▯

5. Conclusions and open problems. We present a reduction converting approximate offline linear optimization problems into approximate online sequential linear optimization problems that hold for *any* approximation algorithm, in both the full-information setting and the bandit setting.

Our algorithm can be viewed as an analogue to Hannan's algorithm for playing repeated games against an unknown opponent. In our case, however, we cannot compute best responses but only approximately best responses.

The problem of obtaining similar results for interesting classes of nonlinear optimization problems remains open.

Appendix. Example where "follow-the-leader" fails. First consider the set $\mathcal{S} = \{1, 2, \ldots, n\}$ and the cost sequence $(1, 1, \ldots, 1)$ (repeated T/n times), $(1, 0, \ldots, 0)$ (repeated T/n times), $(0, 1, 0, \ldots, 0)$ (repeated T/n times),..., $(0, \ldots, 0, 1)$ (repeated T/n times). Notice that a selection of decision, each period, which costs 1 is always a valid ($\alpha = 2$)-approximation to the leader on the previous examples. Moreover, its cost is T while the cost of the best (in fact *every*) $s \in \mathcal{S}$ is $2T/n$, hence giving large α-regret. Unfortunately, adding perturbations of $O(\sqrt{T})$ as in "follow-the-leader" will not significantly improve matters. When $T/n \gg \sqrt{T}$, a choice of decision which costs 1 each period is still an α-approximation for, say, $\alpha = 3$.

Of course, one may be suspicious that no common approximation algorithms would have such peculiar behavior. We now give a similar example based on the standard greedy set cover approximation algorithm A ($\alpha = \log m$) applied to the online set cover problem described earlier. The example has $n/2$ covers of size 2: $S_i = S \setminus S_{n+1-i}$ for $i = 1, 2, \ldots, n$. Furthermore, suppose the sets are of increasing size $|S_i| = \big(0.4 + 0.2\frac{i-1}{n-1}\big)m$ and $|S_i \cup S_j| \leq 0.9\,m$ for all $1 \leq i, j \leq n$, where $i \neq n+1-j$.[8] The sequence of costs (weight) vectors is divided into $n/2$ phases $j = 0, 1, \ldots, n/2-1$, each consisting of $2T/n$ identical cost vectors. In phase $j = 0$, all sets have cost 1. For phase $j = 1, \ldots, n/2-1$ the following hold: the cost of the $2j-1$ sets $S_1, \ldots, S_j$ and $S_{n-j+1}, \ldots, S_n$ are all 1, while the costs of the remaining sets are all 0.

In this example, following the leader with greedy set cover will have an average per-period cost of at least 0.1. In particular, during the first 10% of any phase $j \geq 1$, either greedy algorithm's first choice will be S_{n-j}, in which case its second choice will be S_j (because any other set covers at most 90% of the remaining items, and S_j's cost so far is at most 10% more than that of any other set), or greedy's first choice will be one of $S_{n-j+1}, \ldots, S_n$; in either case it pays at least 1 during that period. Hence, "following-the-leader" pays at least $0.1 + \frac{19}{5}n$ in expectation on average, while the cover $S_{n/2} \cup S_{n/2+1}$ has an average cost of only $4/n$, which is far from matching greedy's $\alpha = \log m$ approximation ratio (for $n = \theta(m)$).

Also note that perturbations on the order of $O(\sqrt{T})$ will not solve this problem. It would be very interesting to adapt Hannan's approach to work for approximation algorithms, especially because it is more efficient than our approach. However, we have not found a solution that works across problems.

REFERENCES

[1] J. Abernathy, E. Hazan, and A. Rakhlin, *Competing in the dark: An efficient algorithm for bandit linear optimization*, in Proceedings of the 21st Annual Conference on Learning Theory (COLT), 2008.

[2] B. Awerbuch and R. Kleinberg, *Adaptive routing with end-to-end feedback: Distributed learning and geometric approaches*, in Proceedings of the 36th ACM Symposium on Theory of Computing (STOC), 2004.

[3] P. Bartlett, V. Dani, T. P. Hayes, S. M. Kakade, A. Rakhlin, and A. Tewari, *High probability regret bounds for bandit online optimization*, in Proceedings of the 21st Annual Conference on Learning Theory (COLT), 2008.

[4] A. Blum and M.-F. Balcan, *Approximation algorithms and online mechanisms for item pricing*, Theory Comput., 3/9 (2007), pp. 179–195.

[5] R. Carr and S. Vempala, *Randomized metarounding*, Random Structures Algorithms, 20 (2002), pp. 343–352.

[8]To design such a collection of sets (for even n and $m = 5(n-1)$), take S_i to be a uniformly random set of the desired size m for $i = 1, \ldots, n/2$, and S_{n+1-i} to be its complement. It is not hard to argue that, with high probability, the randomized construction obeys the stated properties.

[6] D. Chakrabarty, A. Mehta, and V. Vazirani, *Design is as easy as optimization*, in 33rd International Colloquium on Automata, Languages and Programming (ICALP), 2006.
[7] V. Dani and T. P. Hayes, *Robbing the bandit: Less regret in online geometric optimization against an adaptive adversary*, in Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2006.
[8] V. Dani, T. P. Hayes, and S. M. Kakade, *The price of bandit information for online optimization*, in Advances in Neural Information Processing Systems 20 (NIPS), 2007.
[9] J. Dunagan and S. Vempala, *A simple polynomial-time rescaling algorithm for solving linear programs*, Math. Program., 114 (2008), pp. 101–114.
[10] A. D. Flaxman, A. T. Kalai, and H. B. McMahan, *Online convex optimization in the bandit setting: Gradient descent without a gradient*, in Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2005, pp. 385–394.
[11] M. X. Goemans and D. P. Williamson, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, J. ACM, 42 (1995), pp. 1115–1145.
[12] J. F. Hannan, *Approximation to Bayes risk in repeated play*, in Contributions to the Theory of Games, M. Dresher, A.W. Tucker, and P. Wolfe, eds., Vol. 3, Princeton University Press, Princeton, NJ, 1957, pp. 97–139.
[13] A. Kalai and S. Vempala, *Efficient algorithms for online decision problems*, J. Comput. System Sci., 71 (2005), pp. 291–307.
[14] H. B. McMahan and A. Blum, *Online geometric optimization in the bandit setting against an adaptive adversary*, in Proceedings of the 17th Annual Conference on Learning Theory (COLT), 2004.
[15] H. Robbins, *Some aspects of the sequential design of experiments*, Bull. Amer. Math. Soc. (N.S.), 55 (1952), pp. 527–535.
[16] M. Zinkevich, *Online convex programming and generalized infinitesimal gradient ascent*, in Proceedings of the 20th International Conference on Machine Learning (ICML), 2003.