2014 IEEE Annual Symposium on Foundations of Computer Science

Chasing Ghosts: Competing with Stateful Policies

Uriel Feige
Weizmann Institute
uriel.feige@weizmann.ac.il

Tomer Koren
Technion and Microsoft Research
tomerk@technion.ac.il

Moshe Tennenholtz
Microsoft Research and Technion
moshet@microsoft.com

***Abstract*—We consider sequential decision making in a setting where regret is measured with respect to a set of *stateful* reference policies, and feedback is limited to observing the rewards of the actions performed (the so called "bandit" setting). If either the reference policies are stateless rather than stateful, or the feedback includes the rewards of all actions (the so called "expert" setting), previous work shows that the optimal regret grows like $\Theta(\sqrt{T})$ in terms of the number of decision rounds T.**

The difficulty in our setting is that the decision maker unavoidably loses track of the internal states of the reference policies, and thus cannot reliably attribute rewards observed in a certain round to any of the reference policies. In fact, in this setting it is impossible for the algorithm to estimate which policy gives the highest (or even approximately highest) total reward. Nevertheless, we design an algorithm that achieves expected regret that is sublinear in T, of the form $O(T/\log^{1/4} T)$. Our algorithm is based on a certain *local repetition lemma* that may be of independent interest. We also show that no algorithm can guarantee expected regret better than $O(T/\log^{3/2} T)$.

I. Introduction

A player is faced with a sequential decision making task, continuing for T rounds. There is a finite set $[n] = \{1, \ldots, n\}$ of actions available in every round. In every round, based on all information observed in previous rounds, the player may choose an action $i \in [n]$, and consequently receives some reward $r \in [0, 1]$ on that particular round. The total reward of the player is the sum of rewards accumulated in all rounds. There are various policies suggested to the player as to how to choose the sequence of actions in a way that would lead to high total reward. Examples of policies can be to play action 2 in all rounds, to play action 2 in odd rounds and action 3 in even rounds, or to start with action 1, play the current action repeatedly in every round until the first round in which it gives payoff less than $1/2$, then switch to the next action in cyclic order, and so on. The number of given policies is denoted by k. A-priori the player does not know which is the better policy. An algorithm of the player is simply a new policy that may be based on the available given policies. For example, the algorithm may be to follow policy number 5 in the first $T/2$ rounds, and play action 3 in the remaining rounds. The regret of the algorithm of the player is the difference between the total payoff of the best given policy to that of the player's algorithm. Our goal is to design an algorithm for the player that has as small regret as possible.

There are many different variations on the above setting, and some have been extensively studied in the past, with two of the most common variations referred to as expert algorithms and bandit algorithms [5, 9, 3]. In this work we study a natural variation that apparently did not receive much attention in the past. We present this variation in its simplest form in Section I-A, and defer discussion of extensions to Section I-F.

I.A. The Stateful Policies Model

We view the sequential decision making problem as a repeated game between a player and an adversary. Before the game begins, the adversary determines a sequence[1] of reward functions $r_{1:T} = (r_1, \ldots, r_T)$, where each function assigns each of the actions in $[n]$ with a reward value in the interval $[0, 1]$. We refer to such adversary as *oblivious*, since the functions $r_{1:T}$ cannot change as a result of the player's actions (as they are chosen ahead of time). On each round t, the player must choose, possibly at random, an action $X_t \in [n]$. He then receives the reward $r_t(X_t)$ associated with that action, and his feedback on that round consists of this reward only; this is traditionally called *bandit feedback*.

The player is given as input a set Π of $k > 1$ *policies*, which are referred to as the *reference policies*. Each policy $\pi \in \Pi$ is a deterministic function that maps the sequence of all previously observed rewards into an action to be played next. For a policy π, we use the notation x_t^π to denote the action played by π on round t, had it been followed from the beginning of the game (note that x_t^π has a deterministic value). The player's goal is to minimize his (expected) *regret* measured with respect to the set of reference policies Π, defined by

$$\mathrm{Regret}_T \;=\; \max_{\pi\in\Pi} \sum_{t=1}^{T} r_t(x_t^\pi) - \mathbf{E}\left[\sum_{t=1}^{T} r_t\big(X_t\big)\right] .$$

We say that the player's regret is non-trivial if it grows sublinearly with T, namely if $\mathrm{Regret}_T = o(T)$.

While regret measures the performance of a specific algorithm on a particular sequence of reward functions, we are typically interested in understanding the intrinsic difficulty

[1] We use the notation $a_{s:t}$ as a shorthand for the sequence $(a_s, \ldots, a_t)$.

0272-5428/14 $31.00 © 2014 IEEE
DOI 10.1109/FOCS.2014.19

of the learning problem. This difficulty is captured by the game-theoretic notion of *minimax regret*, which intuitively is the expected regret of an optimal algorithm when playing against an optimal adversary. Formally, the minimax regret is defined as the infimum over all player algorithms, of the supremum over all reward sequences, of the expected regret.

In this paper we consider a type of reference policies that we refer to as *stateful policies*, which we define next (see also Fig. 1 for an illustration of this concept).

Definition 1 (stateful policy)**.** A *stateful policy* $\pi = (s_0^\pi, f^\pi, g^\pi)$ over n actions and S states is a finite state machine with state space $[S] = \{1, 2, \ldots, S\}$, characterized by three parameters:

(i) the *initial state* of the policy $s_0^\pi \in [S]$, which is used to initialize the policy before the first round;

(ii) the *action function* $f^\pi : [S] \mapsto [n]$, describing which action to take in a given round, depending on the state the policy is in;

(iii) the *state transition function* $g^\pi : [S] \times [0,1] \mapsto [S]$, which given the current state and the observed reward of the action played in the current round, determines to which state to move for the next round.

The action x_t^π played by a stateful policy π on round t (had π been followed from the beginning of time) can be computed recursively, starting from the given initial state s_0^π, according to

$$\forall\ t \qquad \begin{aligned} x_t^\pi &= f^\pi(s_{t-1}^\pi)\ , \\ s_t^\pi &= g^\pi(s_{t-1}^\pi, r_t(x_t^\pi))\ . \end{aligned}$$

Here, s_t^π represents the state π reaches at the end of round t.

In our setting, we assume that the player is given as input a reference set Π of $k > 1$ stateful policies, each over at most S states. The player may base his decisions on the description of the k reference policies (in particular, the policies can serve as subroutines by his algorithm). Without loss of generality, we shall assume that each policy in Π has exactly S states. Also, for simplicity we assume that policies are *deterministic* (involve no randomization) and *time-independent*: the functions f^π and g^π do not depend on the round number; see Section I-F for extensions of our results to randomized and to time-dependent policies.

Example. We present a detailed example to illustrate the model. Suppose that our player, a driver, faces a daily commute problem, that repeats itself for a very large number T of days. There are three possible routes that he can take, and an action is a choice of route (hence $n = 3$). Each of the three routes can be better than the others on any given day. The reward of the player on a given route in a given day is some number in the range $[0, 1]$ that summarizes his satisfaction level with the route he took (taking into account the time of travel, road conditions, courtesy of other drivers, and so on). The driver learns this reward only after taking the route, and does not know what the reward would have been had he taken a different route. We further assume that the effect of a single driver on traffic experience is negligible: the presence of the driver on a particular route on a given day has no effect of the quality (satisfaction level) of that or any other route on future days.

The driver is told that there is a useful policy for choosing the route in a given day, based only on the reward of the previous day. This policy has three states (hence $S = 3$). The action function f is simply the identity function (in state i take route i). The state transition function g is independent of the current state, and depends only on the reward received. If x denotes the reward received in the current day, then the next state is as follows: $g(x) = 1$ for $|x - \frac{1}{2}| \leq \frac{1}{6}$, $g(x) = 2$ for $|x - \frac{1}{2}| > \frac{1}{3}$, and $g(x) = 3$ otherwise. The only part not specified by the policy is the initial state s_0 (which route to take on the first day). Hence effectively there are three reference policies, different only on their initial state, and thus $k = 3$.

The beauty of the policy, so the player is told, is that if he gets the initial state right and from then on follows the policy blindly, his overall satisfaction is guaranteed. Not knowing which is the better reference policy, does the player have a strategy that guarantees sublinear regret (in T) against the best of the three reference policies? If so, how low can this regret be guaranteed to be?

The kind of policies considered in our example above is perhaps the weakest type of a stateful policy, one that we refer to as a *reactive policy*.

Definition 2 (reactive policy)**.** A *reactive policy* π over n actions (with 1-lookback) is specified by an initial action $x_1^\pi \in [n]$ to be played in the first round of the game, and by a function $\pi : [0,1] \mapsto [n]$ that maps the observed reward of the action played in the current round to an action to be played on the next round.

A reactive policy simply reacts to the last reward it receives as feedback and translates it into an action to be played on the next round. A reactive policy can be seen as a special type of a stateful policy with $S = n$ states, if we identify each of the sets $\pi^{-1}(i) \subseteq [0,1]$ with a unique state $i \in [n]$. In this view, the action function f^π is simply the identity function, and the state transition function g^π is independent of the current state (and maps a reward r to the state i if $r \in \pi^{-1}(i)$). See also Fig. 1 for a visual description of the reactive policies used in our example.

I.B. Main Results

We now state our main results, which are upper and lower bounds on the expected regret in the stateful policies model.

Theorem 3. *For any given $k, S \geq 1$, there is an algorithm for the player that guarantees sublinear expected regret with respect to any reference set Π of k stateful policies over S*

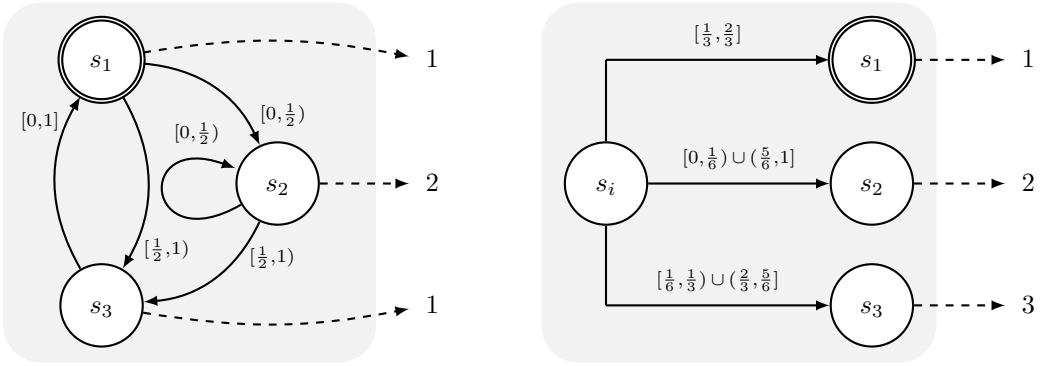

Figure 1: (Left) A stateful policy over $n = 2$ actions with $S = 3$ states, with s_1 being the initial state. The labels on the edges between states indicate the set of rewards that trigger the corresponding transition (which is the role of the function g^π). The dashed arrows depict the function f^π, that assigns each state with an action. (Right) A reactive policy over $n = 3$ actions, considered in the example of Section I-A. The state s_i is a placeholder that stands for each of s_1, s_2, s_3 and shows the outgoing transitions that are common to all 3 states.

states. Specifically, for any set Π *and any oblivious sequence of reward functions, Algorithm 3 given in Section II-C achieves an* $O\left(\sqrt{kS}\cdot\frac{T\log\log T}{\log^{1/4}T}\right)$ *upper bound over the expected regret with respect to* Π.

Though the regret achieved in Theorem 3 is sublinear, it is only slightly so. Unfortunately, this is unavoidable.

Theorem 4. *No player algorithm can guarantee expected regret better than* $O(T/\log^{3/2}T)$ *with respect to any set of* $k = 3$ *reference policies over* $S = 3$ *states and* $n = 3$ *actions, not even if the reference policies are all reactive (as in Definition 2). Moreover, this negative result holds in the commute example given in Section I-A.*

For proving the above bounds, it will be convenient for us to first obtain upper and lower regret bounds in a simplified model we call *the hidden bandit*. This model precisely captures the main difficulties associated with the stateful policies setting, and may be of independent interest. Our results in the hidden bandit setting will be stated after we establish the required definitions in Section I-D.

I.C. Discussion and Related Work

A unifying paradigm for virtually all previous sequential optimization algorithms, whether in the expert or bandit setting, is the following. As the rounds progress, the algorithm "learns" which arm had the better past performance (in the expert setting the algorithm observes all arms, in the bandit setting the algorithm uses an "exploration and exploitation" procedure), and then plays this arm (either deterministically or with high probability). This paradigm is not suitable for our stateful policies model (with bandit feedback), as there is no way by which the algorithm can learn the identity of the best reference policy, even if this reference policy gives reward 1 in every round and all other reference policies give reward 0 in every round. This difficulty stems from the fact that reference policies might differ only in their initial state, and their identity is lost because the player cannot track the state evolution of policies, due to the bandit nature of the feedback. (This aspect will become more evident in the proof of Theorem 4.)

Several variants of our model have been extensively studied in the past. However, to the best of our knowledge, our results constitute the first known example of a learning problem where the minimax regret rate is of the form $\Theta(T/\mathrm{polylog}(T))$. For this reason, we believe that the problem we consider is substantially different from previously studied, seemingly related sequential decision problems.

The full-feedback analog of our setting is widely known to be captured by the so-called "experts" framework, and has been studied under the name of "simulatable experts" [4]. Basically, when the player observes the rewards of all actions he is able to "simulate" each of his contending policies and keep track of their cumulative rewards. Hence, we can treat each policy as an independent expert and use standard online learning techniques (such as the weighted majority algorithm) to obtain $O(\sqrt{T})$ regret in this setting. Consequently, we exhibit an *exponential gap* between the minimax regret rates of the full-feedback and bandit-feedback variants of the problem[2]. As far as we know, this is the first evidence of such gap to date: the only previously known gap is in the case of the multi-armed bandit problem with switching costs, where the minimax regret rates are $\Theta(\sqrt{T})$ and $\widetilde{\Theta}(T^{2/3})$ in the full-feedback and bandit-feedback versions, respectively [2, 6].

Further discussion of related work is deferred to the full version of the paper [8].

I.D. The Hidden Bandit Problem

In this section we present a setting that we shall refer to as *the hidden bandit problem*, which captures the main difficulties associated with the stateful policies model. It will

[2]We say that the gap between the achievable rates is exponential, since the average (per-round) regret decays like $1/\mathrm{polylog}(T)$ in the bandit case, while in the full-information case it decays like $1/\sqrt{T}$.

be convenient for us to first obtain results in the hidden bandit model, and then translate them to the stateful policies model.

To motivate the hidden bandit problem, let us discriminate between two different modes a player in the stateful policies model may be in, at any given round: the "good mode", in which the algorithm is following the best reference policy in its correct state, and the "bad mode" in which the algorithm is doing something else (i.e., following other reference policy or executing a sequence of actions that do not correspond to any reference policy). Inevitably, the player is not aware of his current mode and is unable to switch between the modes deterministically. However, if at some point in time the player is told that he is in the "good mode", then from that point onwards he can replicate the actions of the best policy by observing its rewards and emulating its state transitions, and remain in the same mode.

Roughly, the hidden bandit problem can be described as a multi-armed bandit problem with two arms, the *reference arm* and the *decoy arm*, that correspond to the "good mode" and the "bad mode" in the stateful policies model, respectively. Unlike standard multi-armed bandit problems, a key aspect of this problem is that in any given round the player does not know which of the arms he is currently pulling. Accordingly, the player is not able to select which arm to pull on each round; rather, he can only choose whether to *stay* on the current arm or to *switch* to the other arm with some probability. These aspects capture the difficulties in the stateful policies model, in which once the player leaves a certain policy, attempting to return to that policy involves guessing correctly the policy's internal state, an aspect that a player is not sure of.

The model: We now turn to the formal description of the hidden bandit model. There are two parameters associated with the hidden bandit model. One is T, the number of rounds, and the other is p, a parameter in the range $0 < p < 1$. There are two arms, arm 0 and arm 1, that will be referred to as the *reference arm* and the *decoy arm*, respectively. At each round, the player has only two possible actions available:

- stay: stays on the same arm on which the player entered the round;
- switch: switches to arm 1 if the player entered the round on arm 0; otherwise, switches to arm 0 with probability p, and stays on arm 1 with probability $1-p$.

The dynamics of the switch action can be seen as a two-state Markov chain, with states corresponding to the arms. Initially, prior to round 1, the player is placed on one of the arms at random, being on arm 0 with probability $\frac{p}{1+p}$ and on arm 1 with probability $\frac{1}{1+p}$. This initial probability distribution is the stationary distribution with respect to the randomized switch action defined above. Hence, any sequence of actions (either stay or switch) of the player gives rise to a sequence of random variables $X_{1:T}$, where $X_t \in \{0,1\}$ indicates which arm is pulled by the player on round t. Even though at each round the player is pulling some arm, *the player cannot observe on which arm he is playing*. In other words, the sequence $X_{1:T}$ is *not observable* by the player.

On each round $t = 1, \ldots, T$, the adversary assigns a reward to each arm. We let $r_t(i) \in [0,1]$ denote the reward of arm i on round t. The rewards of the reference arm are set by the adversary in an *oblivious* way, before the game begins. The rewards of the decoy arm are set by the adversary in an *adaptive* way as the game progresses: at every round t, the reward of arm 1 can be based on the entire history of the game up to round t. The feedback to the player on round t in which arm X_t is played consists only of the reward $r_t(X_t)$, and the player does not get to observe the reward of the other arm on that round.

The goal of the player is to minimize his expected *regret*, which is computed only with respect to the *reference arm*, namely $\mathrm{Regret}_T = \sum_{t=1}^{T} r_t(0) - \mathbf{E}\left[\sum_{t=1}^{T} r_t(X_t)\right]$, where the expectation on the right-hand side is taken with respect to the randomization of the switch actions, as well as to the internal random bits used by the player.

Remark: The fact that the adversary can set the rewards on the decoy arm in an adaptive manner will allow us to simulate any execution in the stateful policies model by an execution in the hidden bandit model. Consequently, all positive results (algorithms with low regret) that we shall prove in the hidden bandit model will transfer easily to the stateful policies model (basically, by setting $p = 1/(Sk)$, where k is the number of reference policies and S is the maximum number of states that a policy might have). On the other hand, it might not be true that negative results in the hidden bandit model transfer to the stateful policies model. Nevertheless, our negative results for the hidden bandit model will be obtained with an *oblivious adversary* (which is oblivious not only on the reference arm but also on the decoy arm), and consequently will transfer to the stateful policies model.

Results: We now present our results for the hidden bandit problem, that we later show how to translate into the corresponding upper and lower bounds in the stateful policies model.

Theorem 5. *For any given $0 < p < 1$, there is an algorithm for the player in the hidden bandit setting that guarantees sublinear expected regret (in T). Specifically, Algorithm 2 presented in Section II-B achieves an expected regret of $O\left(\frac{1}{\sqrt{p}} \cdot \frac{T \log\log T}{\log^{1/4} T}\right)$ over any sequence of reward functions.*

Theorem 6. *For $p = \frac{1}{2}$, no algorithm for the player in the hidden bandit setting guarantees expected regret better than $O(T/\log^{3/2} T)$, not even if the adversary uses an oblivious strategy on both arms.*

There is a gap between the upper bound of Theorem 5 and the lower bound of Theorem 6, that translates into a gap between our main upper and lower bounds of Theorems 3 and 4. In some natural special cases, we are able to close this gap. We say that an adversary is *consistent* if there is a fixed offset $0 < \Delta \leq 1$ such that in every round t, $r_t(0) - r_t(1) = \Delta$. Say that the player's algorithm is *semi-Markovian* if the choice of action taken at any given round depends only on the sequence of rewards obtained since the last switch action. (See exact definitions in [8].)

Theorem 7. *In the hidden bandit setting, if the player's algorithm is required to be semi-Markovian and the adversary is required to be consistent, then there is an algorithm achieving expected regret $O(T/\log T)$, and this is best possible up to constants (that may also depend on p).*

We remark that we actually prove a slightly stronger statement than that of Theorem 7: for the positive results a *Markovian* algorithm suffices, for the negative results a *consistent* adversary suffices. See [8] for more details.

I.E. Our Techniques and Additional Related Work

Our algorithm in the proofs of Theorem 5 and Theorem 3 is based on a principle that to the best of our knowledge has not been used previously in sequential optimization settings. This is the *local repetition lemma* which will be explained informally here, and addressed formally in Section II-A (see Lemma 11).

In the hidden bandit setting, suppose first that the sequence of rewards that the adversary places on the reference arm is *repetitive*—the same reward r on every round. If the player knows that the reference arm is repetitive, it should not be difficult for the player to achieve sublinear regret, even if he does not know what r is. He can start with an *exploration phase* (occasional switch requests embedded in sequences of stay actions) that will alert him to repeated patterns of r values in-between two switches. Thereafter, in an *exploitation phase*, whenever the player gets a reward below r, he will ask for a switch. The only way the decoy arm can cause the player not to reach the reference arm is by offering rewards higher than r, but getting rewards higher than r on the decoy arm causes no regret. (The above informal argument is made formal in the proof of Theorem 5.)

The above argument can be extended (with an $O(\epsilon T)$ loss in the regret) to the case that the rewards on the reference arm are *ϵ-repetitive*, namely, in the range of $r \pm \epsilon$ for some r. Suppose now that given some integer $d < T$, the reference arm is not ϵ-repetitive, but only *(d, ϵ)-locally repetitive*, in the following sense: starting at any round that is a multiple of d, the sequence of rewards on the d rounds that follows is ϵ-repetitive. A (d, ϵ)-locally repetitive sequence need not be ϵ-repetitive—it can change values arbitrarily every d rounds. However, if d is sufficiently large (compared to $1/p$ in the hidden bandit setting), the player should be able to achieve small regret, by breaking the sequence of length T to T/d blocks of size d, and treating each block as an ϵ-repetitive sequence.

But what happens if the rewards on the reference arm are not (d, ϵ)-repetitive? Then we can use a notion of *scales*. For $0 \leq \ell < \log_d T$, the scale-ℓ version of a sequence of length T is obtained by bunching together groups of d^ℓ consecutive rounds into one super-round, and making the reward of the super-round equal to the average of the rewards of the rounds it is composed of. The player in the hidden bandit setting may choose a random scale ℓ, in hope that in this scale the resulting sequence of super rounds is (d, ϵ)-repetitive. It turns out this approach works. This is a consequence of the *local repetition lemma* that we state here informally.

Lemma 11 (Local repetition lemma, informal statement)**.** *For every choice of integer $d \geq 2$ and $0 < \epsilon, \delta < 1$, if T is sufficiently large (as a function of d, ϵ and δ), then for every string in $\sigma \in [0,1]^T$, in almost all scales (say, a fraction of $1 - \delta$) the resulting sequence is almost (d, ϵ)-repetitive (i.e., only a δ fraction of the blocks fail to be ϵ-repetitive).*

We are not aware of a previous formulation of the local repetition lemma. However, it has connections to results that are well known in other contexts. We briefly mention several such connections, without attempting to make them formal. The regularity lemma of Szemerédi asserts that every graph has some "regular" structure. Likewise, the local repetition lemma asserts that every string has some "regular" (in the sense of being nearly repetitive) structure. Our proof for the local repetition lemma follows standard techniques for proving the regularity lemma, though is easier (because strings are objects that are less complicated than graphs). An alternative proof for the local repetition lemma can go through martingale theory (e.g., through the use of martingale upper-crossing inequalities). The relation of our setting to that of martingales is that the sequence of values observed when going from a super round in the highest scale all the way down to a random round in smallest scale is a martingale sequence. Yet another related topic is Parseval's identity for the coefficients of Fourier transforms. It gives an upper bound on the sum of all Fourier coefficients, implying that most of them are small. This means that a random scale a sequence of values has small Fourier coefficients, and small Fourier coefficients correspond to not having much variability at this scale.

Our lower bound of Theorem 6 is based on a construction that was used by Dekel et al. [6] for proving lower bounds on the regret for bandit settings with switching costs. The construction is a full binary tree with T leaves that correspond to the rounds, in which each edge of the tree has a random reward, and the reward at a leaf is the sum of rewards along the root to leaf path. The reward on the decoy arm is identical to that of the reference arm, except for a

constant offset, which on the one hand should not be too large so that the player cannot tell when he is switching between arms, and on the other hand should not be too small as it determines the regret. In the context of Dekel et al. [6], such a construction results in a regret of $\Omega(T^{2/3}/\log T)$. In our context, a similar construction gives a much higher, almost linear lower bound. We remark that our modification of this randomized construction share similarities with a construction used by Dwork et al. [7] to obtain positive results in a different context, that of *differential privacy*. (The inability of the player to distinguish between the reference arm and the decoy arm is analogous to keeping the value of an offset "differentially private".)

The upper bound in Theorem 7 is based on a simple randomized algorithm that in every round asks for a switch with probability that is exponential in the negative of the reward of that particular round. The proof that this algorithm has low regret (when the adversary is consistent) is based on showing that the expected fraction of rounds spent on the decoy arm is exponential in the (negative) offset of the decoy arm compared to the reference arm.

The lower bound in Theorem 7 (against semi-Markovian algorithms) is based on the adversary choosing at random a fixed reward on the reference arm and a fixed smaller reward on the decoy arm. Natural distributions for choosing these two rewards only lead to a regret that behaves roughly like $\Omega(T/\log^{3/2} T)$. To get the matching lower bound of $\Omega(T/\log T)$ we use a distribution similar to the distribution of queries that was used in work of Raskhodnikova [11] on monotonicity testing with a small number of queries.

I.F. Extensions of Our Upper Bound

We discuss a few simple extensions of the basic model presented in Section I-A.

Time-dependent policies: In our stateful policies model, reference policies were assumed to be time independent. We may also consider a model in which reference policies can be time dependent (the functions f^π, g^π have an additional input which is the round number). Our lower bound (Theorem 4) is proved with respect to time independent reference policies, and hence holds without change when reference policies can be time dependent. Our upper bound (Theorem 3) also holds without change when reference policies are time dependent—nothing in the proof of Theorem 3 requires time independence.

Randomized policies: In our stateful policies model, reference policies were assumed to be deterministic. We may also consider a model in which reference policies can be randomized (the functions f^π, g^π have access to random coin tosses). Our lower bound (Theorem 4) is proved with respect to deterministic reference policies, and hence holds without change when reference policies can be randomized. For the upper bound, there are two natural ways of evaluating the regret. One, less demanding, is against the expected total reward of the reference policy with highest total expected reward. The other, more demanding, is against the expectation of the realized maximum of the total rewards of the reference policies. (That is, one runs each one of the reference policies using independent randomness, and observes which policy achieves the highest reward.) Our upper bound (Theorem 3) extends to randomized reference policies, even under the more demanding interpretation—one simply fixes for each reference policy all its random coin tosses in advance, thus making it deterministic, and then Theorem 3 applies with no change.

Stateful and reactive adversaries: One of the motivations of the current study was to consider also stateful adversaries, and not just stateful policies. For a stateful adversary, the reward at a given round can depend not only on the action taken by the player, but also on the entire history of the game up to that round (via some state variable that the adversary keeps and updates after every round). In general, it is hopeless to attain sublinear regret in such settings (for example, the action taken in the first round might determine the rewards in all future rounds, and then one mistake by the player already gives linear regret). However, our positive results do extend to a certain class of stateful adversaries, for which the reward received at any round is a function of the actions of the player on that and the ℓ previous rounds (for some fixed ℓ). We refer to this class as *reactive adversaries*, in analogy to our notion of reactive policy, though it has been studied in the literature under the names "loss functions with memory" [10] and "bounded memory adaptive adversary" [1]. See [8] for more details.

II. Proofs

II.A. The Local Repetition Lemma

In this section we formulate and prove the local repetition lemma, which is a key lemma for the proof of Theorem 5. As this lemma may have other applications, we use a generic terminology that is not specific to our sequential decision models. In the notation of the local repetition lemma, a sequence will be referred to as a string, its length will typically be denoted by n (rather than T), and the entries of the string (which will still have values in $[0,1]$) will be referred to as characters rather than rewards. Hence, strings are concatenation of individual characters, where the value of a character is a real number in the range $[0,1]$. However, it will be convenient for us to sometimes view a string as a concatenation of substrings. Namely, each entry of the string might be a string by itself, and the whole string is a concatenation of these substrings. We may apply this view recursively, namely, the entries of each substring might also be substrings rather than individual characters. The notation that we introduce below is flexible enough to encompass this view.

For arbitrary n, given a string $s \in [0,1]^n$, x_s denotes its average value. Using $s(i)$ to denote the ith entry of s, and

using $x_{s(i)}$ to denote the value of this entry, we thus have $x_s = \frac{1}{n}\sum_{i=1}^{n} x_{s(i)}$. This notation naturally extends to the case that s is not a string of characters, but rather a string of n substrings, in which each substring $s(i)$ is by itself a string of m characters (same m for every $1 \leq i \leq n$). In this case, $x_{s(i)}$ is the average value of string $s(i)$, and the expression $\frac{1}{n}\sum_{i=1}^{n} x_{s(i)}$ still correctly computes x_s.

As a rule, whenever we view a string as being composed of substrings, all these substrings will be of exactly the same length.

Definition 8 (repetitive string)**.** Let n be a multiple of d. Consider a string $s \in [0,1]^n$, viewed as a concatenation of d substrings, $s(1), \ldots, s(d)$, each in $[0,1]^{n/d}$. Given $\epsilon > 0$, we say that s is (d,ϵ)-*repetitive* if for every i we have $|x_s - x_{s(i)}| \leq \epsilon$.

A key aspect of our approach is that we shall typically not consider the string as a whole, but rather consider only a *local* portion of the string, namely, a substring. Moreover, the size of the local portion depends on the level of resolution at which we wish to view the string. Consequently, we endow the string with a probability distribution over its substrings, as in Definition 9.

Definition 9 (d-sampling)**.** Let n be a power of d, say $n = d^k$. A *d-sampling* of a string $s \in [0,1]^n$ proceeds as follows. First a value ℓ (for *level*) is chosen uniformly at random from $\{0, \ldots, k-1\}$. Then s is partitioned into d^ℓ consecutive substrings, each of length $d^{k-\ell}$. Thereafter, one of these substrings is chosen uniformly at random, and declared the result of the sampling.

The result of d-sampling is always a string whose length is divisible by d, and hence compatible in terms of length with the requirements of Definition 8.

Remark. In Definition 9 we assume that n is a power of d. We shall make similar simplifying assumptions throughout this section. However, our work easily extends to cases that n is not a power of d. We explain how to do this in the context of d-sampling. Let k be largest such that $d^k \leq n$. With probability d^k/n choose the prefix of length d^k of s and on it do d-sampling as in Definition 9. With the remaining probability $1 - d^k/n$ choose the suffix of length $n - d^k$ of s, and recursively partition it into a prefix and suffix as above, applying Definition 9 only to the prefix. When the suffix becomes shorter than d, stop (this suffix can be discarded from s without affecting our results).

We can now state the key definition for this section.

Definition 10 (locally-repetitive string)**.** Let n be a power of d, and consider a string $s \in [0,1]^n$. Given $\epsilon, \delta > 0$, we say that s is (d,ϵ,δ)-*locally-repetitive* if with probability at least $1-\delta$, a random substring of s sampled using d-sampling (as in Definition 9) is (d,ϵ)-repetitive (as in Definition 8).

The main result of this section is the following.

Lemma 11 (Local repetition lemma)**.** *Let d be a positive integer, and $\epsilon, \delta > 0$. Then for every $n > d^k$ where $k = \frac{d}{4\epsilon^2\delta}$, every string $s \in [0,1]^n$ is (d,ϵ,δ)-locally repetitive.*

Proof: For simplicity, we shall assume that $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$ are integers. Let s be a string in $[0,1]^n$ with $n = d^k$. We say that a substring v is *aligned* if its location in s is such that it may be obtained as a result of d-sampling. Observe that if v is aligned, then it is a concatenation of d equal length strings $v(1), \ldots, v(d)$, each of which is aligned as well. Recall that we refer to ℓ in Definition 9 as the *level*. We use the notation $v \in \ell$ to say that v is aligned, and moreover, v is in level ℓ with respect to d-sampling. Define the *variability* of level $0 \leq \ell \leq k$ to be $V_\ell = \frac{1}{d^\ell}\sum_{v \in \ell}(x_v)^2$.

Proposition 12. *We have $V_k - V_0 \leq \frac{1}{4}$.*

V_ℓ is monotonically nondecreasing with ℓ, because for a given aligned string v with substrings $v(1), \ldots, v(d)$, we have that $x_v = \frac{1}{d}\sum_{i=1}^{d} x_{v(i)}$, and the square of an average is never larger than the average of the squares. For aligned strings v that are not (d,ϵ)-repetitive, the following proposition shows that there is a noticeable increase in variability in the next level.

Proposition 13. *If v is an aligned string that is not (d,ϵ)-repetitive, then $\frac{1}{d}\sum_{i=1}^{d}(x_{v(i)})^2 > (x_v)^2 + \frac{\epsilon^2}{d}$.*

Let δ_ℓ be the conditional probability that given that the d-sampling procedure sampled level ℓ, the substring v sampled is not (d,ϵ)-repetitive. Hence $\delta = \frac{1}{k}\sum_{\ell=0}^{k-1}\delta_\ell$. Then applying Proposition 13 level by level implies $V_k - V_0 > \frac{\epsilon^2}{d}\sum_{\ell=0}^{k-1}\delta_\ell = \frac{k\delta\epsilon^2}{d}$. Contrasting this with Proposition 12 gives $\frac{k\delta\epsilon^2}{d} < \frac{1}{4}$, implying that $\delta < \frac{d}{4\epsilon^2 k}$. ∎

The full proof of the lemma can be found in [8], where we also show that its guarantee is essentially optimal.

II.B. Upper Bound for Hidden Bandits

In this section we present an algorithm for the hidden bandit problem whose worst-case expected regret is sublinear. Our algorithm exploits the fact that the reward sequence of the reference arm, whose values are set in an oblivious manner by the adversary, is (d,ϵ,δ)-locally repetitive (see Definition 10) for appropriately chosen values of d, ϵ, δ, as implied by the local repetition lemma. Hence, it would be instrumental to first consider the simpler case where the reference sequence is in fact (d,ϵ)-repetitive (see Definition 8).

When the reference sequence is (d,ϵ)-repetitive, we propose an algorithm, described in Algorithm 1, which is based on a simple first-explore-then-exploit strategy. The algorithms begins with an exploration phase (Phase I), where it tries to hit the reference arm at least once and obtain an estimate of its reward, which is almost constant at the appropriate scale. Then, in the exploitation phase (Phase

II), the algorithm repeatedly asks for a switch whenever the observed rewards drops below the top estimated rewards obtained in Phase I. Eventually, since the reference arm is (d,ϵ)-repetitive, the algorithm should stabilize on that arm no matter what the rewards on the decoy arm are.

- let $m = \frac{1}{p}\log\frac{1}{\epsilon}$
- **Phase I:** for $i = 1,\ldots,m$: stay on chosen arm for T/d rounds and let $\bar{r}_i$ be the average of the observed rewards, then switch once
- sort the averages $\bar{r}_1,\ldots,\bar{r}_m$ in descending order to obtain $\bar{r}_1 \geq \ldots \geq \bar{r}_m$
- **Phase II:** initialize $i = 1$, $s = 0$ and repeat (until T rounds have elapsed):
  - stay for T/d rounds and let $\bar{r}$ be the average of the observed rewards
  - if $\bar{r} < \bar{r}_i - 2\epsilon$, switch once and update $s \leftarrow s+1$
  - if $s \geq m$, update $i \leftarrow i+1$ and reset $s = 0$

Algorithm 1: Algorithm for (d,ϵ)-repetitive reference sequences.

The following lemma shows that for small values of ϵ, if d is large enough as a function of ϵ then the expected regret of Algorithm 1 is not large (the proof is deferred to [8]).

Lemma 14. *Assume that the reward sequence of the reference arm is (d,ϵ)-repetitive, with $d \geq \frac{1}{p^2\epsilon}\log^2\frac{1}{\epsilon}$. Then the expected regret of Algorithm 1 is at most $8\epsilon T$.*

Our general algorithm, that works for any reference sequence, is described in Algorithm 2. The algorithm invokes Algorithm 1 above as a subroutine on a randomly-chosen block size, exploiting the locally-repetitive structure guaranteed by the local repetition lemma.

- set $\epsilon = \frac{1}{\sqrt{p}}\cdot\frac{\log\log T}{\log^{1/4} T}$, $d = \frac{1}{p^2}\log^2\frac{1}{\epsilon}$
- choose block size $b = d^i$, where i is chosen uniformly at random from $\{1,\ldots,\lfloor\log_d T\rfloor\}$
- for $i = 1,\ldots,T/b$: invoke Algorithm 1 on a block of size b with parameters d,ϵ,p,b

Algorithm 2: Algorithm for the hidden bandit problem.

We are now ready to prove Theorem 5, which gives an upper bound over the expected regret of Algorithm 2. The proof uses the local repetition lemma (Lemma 11).

Proof of Theorem 5: Set $d = \frac{1}{p^2\epsilon}\log^2\frac{1}{\epsilon}$, $\delta = \epsilon$, $k = \frac{d}{4\epsilon^3}$ in Lemma 11, which then states that any sequence of length at least $T_\epsilon = d^k$ is (d,ϵ,ϵ)-locally repetitive. It is not hard to verify that for $T \geq \Omega(1)$ and our choice of ϵ one has $T_\epsilon \leq T$, which means that the reward sequence of the reference arm is (d,ϵ,ϵ)-locally repetitive. Since b was chosen uniformly at random, this means that each b-aligned block of size b in this reward sequence is (d,ϵ)-repetitive with probability at least $1-\epsilon$.

Now, consider a certain iteration of the algorithm. With probability $1-\epsilon$, the corresponding block in the reference reward sequence is (d,ϵ)-repetitive with $d = \frac{1}{p^2\epsilon}\log^2\frac{1}{\epsilon}$. Hence, according to Lemma 14, following the strategy of Algorithm 1 in this block yields an expected regret of $O(\epsilon b)$. Overall, the expected regret in all T/b blocks is then $O(\epsilon T)$. Using the definition of ϵ concludes the proof. ■

II.C. Upper Bound for Stateful Policies

We now show how our algorithm for the hidden bandit setting can be applied, via a simple reduction, in the stateful policy model. The resulting algorithm is presented in Algorithm 3, that provides implementations of the stay and switch actions of the hidden bandit model. The basic idea is to think of the best performing policy (in hindsight) within the set Π as the reference arm, and let the decoy arm capture all other policies, as well as other action paths that do not correspond to any policy.

- choose a policy $\pi_1 \in \Pi$ and a state $s_1 \in [S]$ uniformly at random
- invoke Algorithm 2 with parameters $p = 1/kS$ and T, and the following implementation of stay and switch:
  - stay on round t: play action $f^{\pi_t}(s_t)$, observe reward r, and update $\pi_{t+1} \leftarrow \pi_t$, $s_{t+1} \leftarrow g^{\pi_t}(s_t,r)$
  - switch on round t: play the action $f^{\pi_t}(s_t)$, then choose a policy $\pi_{t+1} \in \Pi$ and a state $s_{t+1} \in [S]$ uniformly at random

Algorithm 3: Algorithm for competing with stateful policies.

We now prove Theorem 3, which provides a regret guarantee for Algorithm 3.

Proof of Theorem 3: Let $\pi^* \in \Pi$ be the best policy in the set Π, namely, the one having the highest total reward in hindsight. For all $t = 1,2,\ldots,T$, we let $s_t^* \in [S]$ denote the state visited by π^* on round t had it been followed from the beginning of the game. Consider a hidden bandit problem where the reward sequence of the reference arm is the sequence obtained by following the policy π^* throughout the game, and the arm being pulled on round t is given by the random variable $X_t = \mathbb{1}_{\pi_t \neq \pi^* \vee s_t \neq s_t^*}$. The decoy arm models any situation where the algorithm deviates from the policy π^*, and each reward obtained on that arm is possibly a function of the entire history of the game, including even the random bits used by the player. Since the model allows for the decoy arm to be completely arbitrary, we do not precisely specify the rewards associated with that arm. The claimed regret bound would then follow from Theorem 5 once we verify that the implementations of the stay and switch actions are correct, namely:

(i) if $X_t = 0$ (i.e., the algorithm is on the reference arm on round t), then choosing stay ensures that $X_{t+1} = 0$;

(ii) if $X_t = 1$ and the algorithm chooses switch then $X_{t+1} = 0$ with probability at least $p = 1/kS$.

Again, since the decoy arm may be completely adversarial, it is not crucial to verify the transitions directed towards it (in particular, the decoy arm might imitate the reference arm in response to a certain action of the algorithm).

To see (i), note that $X_t = 0$ implies $\pi_t = \pi^*$ and $s_t = s_t^*$. In particular, the algorithm picks on round t the same action played by π^* on that round and observes the same reward. Hence, if the algorithm chooses stay then the update $s_{t+1} \leftarrow g^{\pi_t}(s_t, r)$ ensures that $s_{t+1} = s_{t+1}^*$, retaining the algorithm in the correct state on round $t+1$. Next, if $X_t = 1$ which means that the algorithm is not on the reference arm on round t, then by choosing switch the random choice of (π_{t+1}, s_{t+1}) hits the configuration (π^*, s_{t+1}^*) with probability $p = 1/kS$. That is, with probability at least $1/nS$ the algorithm would be on the reference arm on round $t+1$, which proves (ii). ■

Remark. Following the same idea explained in the proof above, it is actually possible to obtain a slightly improved dependence on the number of policies n and save a $n^{1/4}$ factor in the resulting bound, albeit with a more involved algorithm.

II.D. Lower Bound for Hidden Bandits

In this section we prove our lower bound for the hidden bandit problem with $p = \frac{1}{2}$ given in Theorem 6.

In order to prove Theorem 6 we make use of Yao's principle [12], which in our context states that the expected regret of a randomized algorithm on the worst case reward sequence is no better than the expected regret of the optimal deterministic algorithm on any stochastic reward sequence. Hence, Theorem 6 would follow once we establish the existence of a single sequence of stochastic reward functions, $\Gamma_{1:T}$, which is difficult for any deterministic algorithm of the player (in terms of expected regret).

Our construction of the required stochastic sequence $\Gamma_{1:T}$ is based on a variant of the Multi-scale Random Walk stochastic process [6].

Definition 15 (Multi-scale Random Walk [6])**.** Given a sequence $\xi_1, \ldots, \xi_T$ of i.i.d. random variables, the *Multi-scale Random Walk* (MRW) process $W_{0:T}$ is defined recursively by $W_0 = 0$, and

$$\forall\ t \qquad W_t \ = \ W_{\rho(t)} + \xi_t \ , \tag{1}$$

where $\rho(t) = t - 2^{\delta(t)}$, $\delta(t) = \max\{i \geq 0 : 2^i \text{ divides } t\}$.

Our construction, described in Fig. 2, is similar to the one used by Dekel et al. [6], with one crucial difference: instead of using a Gaussian distribution for the step variables $\xi_{1:T}$, we employ a two-sided geometric distribution supported on integer multiples of ϵ (this is a discrete analog of the continuous Laplace distribution). We then use the resulting MRW process $W_{1:T}$ to form a sequence of intermediate reward functions $\widetilde{\Gamma}_{1:T}$, where the reward of arm $x = 0$ is consistently better than that of arm $x = 1$ by a gap of ϵ. The actual reward functions $\Gamma_{1:T}$ are obtained from $\widetilde{\Gamma}_t$ by clipping the reward values to the $[0, 1]$ interval.

- set
$$\epsilon \ = \ \frac{1}{320 \log_2^{3/2} T} \quad \text{and} \quad \gamma \ = \ \frac{1}{4 \log_2 T} \tag{2}$$
- define $W_{1:T}$ to be a MRW process generated according to Eq. (1), where $\xi_{1:t}$ are i.i.d. random variables equipped with the distribution
$$\forall\ n \in \mathbb{Z}\ , \quad \Pr(\xi_t = \epsilon n) \ = \ \tfrac{1-e^{-\gamma}}{1+e^{-\gamma}} \cdot e^{-\gamma|n|} \tag{3}$$
- for $x \in \{0, 1\}$ set
$$\forall\ t \qquad \widetilde{\Gamma}_t(x) \ = \ \tfrac{1}{2} + W_t - \epsilon\, \mathbb{1}_{x \neq 0}\ ,$$
$$\Gamma_t(x) \ = \ \mathrm{clip}\big(\widetilde{\Gamma}_t(x)\big)\ ,$$
where $\mathrm{clip}(r) = \min\{\max\{r, 0\}, 1\}$

Figure 2: An oblivious strategy that forces a regret of $\Omega(T/\log^{3/2} T)$ for any algorithm for the hidden bandit problem with $p = 1/2$.

For this construction, we prove the following lower bound on the performance of any deterministic algorithm that immediately implies Theorem 6.

Theorem 16. *The expected regret of any deterministic player algorithm on the stochastic sequence of reward functions $\Gamma_{1:T}$ defined in Fig. 2 is at least $10^{-4} \cdot T/\log_2^{3/2} T$.*

The proof of the theorem is omitted from this extended abstract, and can be found in [8].

II.E. Lower Bound for Stateful Policies

We now sketch a proof of Theorem 4, using the lower bound proved in Section II-D in the hidden bandit setting. For the formal proof of the theorem, refer to [8].

Proof of Theorem 4 (sketch): Suppose that there is an algorithm A with $o(T/\log^{3/2} T)$ worst-case expected regret. We will show that A can be used to achieve the same expected regret in the hidden bandit model with $p = \frac{1}{2}$, in contradiction to Theorem 6. Given an instance of the hidden bandit problem, we will construct a set of reference reactive policies $\Pi = \{\pi_1, \pi_2, \pi_3\}$ and a randomized construction of reward functions over actions $\{1, 2, 3\}$ that simulates it.

For any two consecutive rounds, we choose a random permutation over the three actions. This defines three disjoint random paths of actions throughout the game, each corresponds to one of the reference policies π_1, π_2, π_3. The reference policies will all share the same action function π and only differ by their initial action on round $t = 1$, where policy π_i begins by playing action i. One of the paths, chosen at random, is the "good" path (that corresponds to the

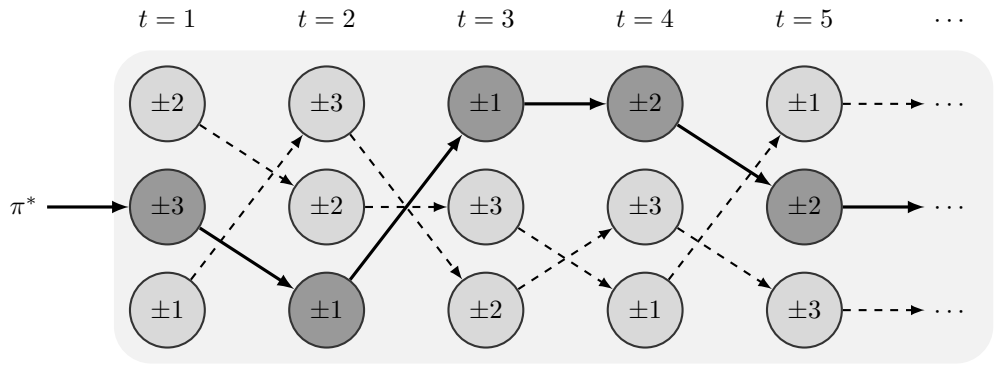

Figure 3: An illustration of the reward functions and policies used in the reduction. The marked path represents the path of the policy π^*, that corresponds to the reference arm in the hidden bandit problem. The absolute value of each rounded reward on one of the paths indicates the next action on that path.

best policy π^*) along which the rewards encountered are the rewards of the reference arm. The other two are the "bad" ones, both assigned with the rewards of the decoy arm. Next, we round the rewards via randomized rounding to either ± 1, ± 2, or ± 3, in a way that keeps the correct expected reward.[3] The type of rounding used for each reward value (i.e., the absolute value of the rounded reward) on each of the paths signals the action function π which is the next action to be played on that path. See Fig. 3 for an illustration of the resulting structure.

Now, we can execute the assumed algorithm A on the construction described above, and translate the sequence of actions it produces into a sequence of stay and switch actions for the underlying hidden bandit problem that achieves $o(T/\log^{3/2} T)$ expected regret. We can do so by emitting a switch action on round t if and only if A deviates from the next action on its current path (as indicated by π) on that round. It is not hard to show that A remains on the same path whenever stay is emitted, and when switch is emitted the probability of switching between good and path paths precisely matches the dynamics of the switch action in the hidden bandit model (see definition in Section I-D). ■

ACKNOWLEDGMENT

Work of Uriel Feige done in part at Microsoft Research, Herzliya, and supported in part by the Israel Science Foundation (grant No. 621/12) and by the I-CORE Program of the Planning and Budgeting Committee and The Israel Science Foundation (grant No. 4/11).

REFERENCES

[1] R. Arora, O. Dekel, and A. Tewari. Online bandit learning against an adaptive adversary: from regret to policy regret. In *Proceedings of the 29th International Conference on Machine Learning (ICML'12)*, pages 1503–1510, 2012.

[2] J.-Y. Audibert, S. Bubeck, et al. Minimax policies for adversarial and stochastic bandits. In *Proceedings of the 22th Annual Conference on Learning Theory (COLT'09)*, pages 217–226, 2009.

[3] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[4] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning and Games*. Cambridge University Press, 2006.

[5] N. Cesa-Bianchi, Y. Freund, D. Haussler, D. P. Helmbold, R. E. Schapire, and M. K. Warmuth. How to use expert advice. *Journal of the ACM (JACM)*, 44(3): 427–485, 1997.

[6] O. Dekel, J. Ding, T. Koren, and Y. Peres. Bandits with switching costs: $T^{2/3}$ regret. In *Proceedings of the 46th Symposium on Theory of Computing (STOC)*, pages 459–467, 2014.

[7] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum. Differential privacy under continual observation. In *Proceedings of the 42nd ACM symposium on Theory of computing*, pages 715–724. ACM, 2010.

[8] U. Feige, T. Koren, and M. Tennenholtz. Chasing ghosts: Competing with stateful policies. *arXiv preprint arXiv:1407.7635*, 2014.

[9] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[10] N. Merhav, E. Ordentlich, G. Seroussi, and M. J. Weinberger. On sequential strategies for loss functions with memory. *Information Theory, IEEE Transactions on*, 48(7):1947–1958, 2002.

[11] S. Raskhodnikova. *Monotonicity testing*. PhD thesis, Massachusetts Institute of Technology, 1999.

[12] A. Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 222–227, 1977.

[3] For simplicity, we construct reward functions that assign reward values in the range $[-3, 3]$ instead of $[0, 1]$; this only affects the constants in the resulting regret bounds.