SIAM J. Comput.
Vol. 46, No. 1, pp. 190–223

CHASING GHOSTS: COMPETING WITH STATEFUL POLICIES[*]

URIEL FEIGE[†], TOMER KOREN[‡], AND MOSHE TENNENHOLTZ[§]

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 experts 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 $O(T/\log^{3/2} T)$.

Key words. sequential decision making, online learning, multiarmed bandit, bandit feedback, stateful policies

AMS subject classifications. 68Q25, 68T05, 68W27

DOI. 10.1137/14100227X

1. Introduction.

A player is faced with a sequential decision making task, continuing for T rounds. There is a finite set $[n] = \{1, \dots, n\}$ 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.

[*]Received by the editors January 2, 2015; accepted for publication (in revised form) October 26, 2015; published electronically February 28, 2017. A preliminary version of these results appeared in *Proceedings of the* 55th *Annual Symposium on Foundations of Computer Science*, 2014.
http://www.siam.org/journals/sicomp/46-1/100227.html

[†]Weizmann Institute of Science, Rehovot, Israel (uriel.feige@weizmann.ac.il). The research of this author was partially done at Microsoft Research, Herzliya, and was supported in part by the Israel Science Foundation (grant 621/12) and by the I-CORE Program of the Planning and Budgeting Committee and the Israel Science Foundation (grant 4/11).

[‡]Technion–Israel Institute of Technology, Haifa, Israel, and Microsoft Research, Herzliya, Israel (tomerk@technion.ac.il).

[§]Technion–Israel Institute of Technology, Haifa, Israel (moshet@ie.technion.ac.il). The research of this author was partially done at Microsoft Research, Herzliya.

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 "experts" algorithms and "bandit" algorithms [4, 14, 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 1.1 and defer discussion of extensions to section 1.7.

1.1. 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 of reward functions $r_{1:T} = (r_1, \dots, r_T)$,[1] 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 π been followed from the beginning of the game. Given that the sequence of reward functions $r_{1:T}$ is already fixed, 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 nontrivial if it grows sublinearly with T, namely, if $\mathrm{Regret}_T = o(T)$.

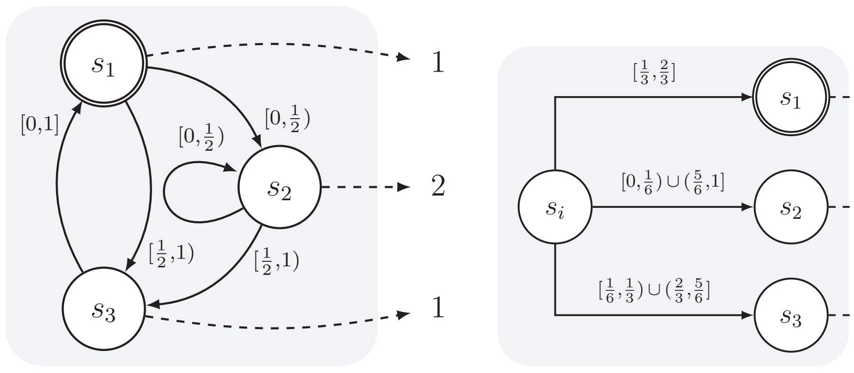
While regret measures the performance of a specific algorithm on a particular sequence of reward functions, we are typically interested in understanding the intrinsic difficulty of the learning problem. This difficulty is captured by the game-theoretic notion of *minimax regret*, which intuitively is 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 Figure 1 for an illustration of this concept).

DEFINITION 1.1 (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 $[S] = \{1, 2, \dots, S\}$, 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;*

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



s_1
1
[0,1]
$[0,\frac{1}{2})$
$[0,\frac{1}{2})$
s_2
2
$[\frac{1}{2},1)$
$[\frac{1}{2},1)$
s_3
1
$[\frac{1}{3},\frac{2}{3}]$
s_1
1
s_i
$[0,\frac{1}{6})\cup(\frac{5}{6},1]$
s_2
2
$[\frac{1}{6},\frac{1}{3})\cup(\frac{2}{3},\frac{5}{6}]$
s_3
3

Fig. 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* 1.1. *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 three states.*

(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

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

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 1.7 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 of 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 1.2 (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 Figure 1 for a visual description of the reactive policies used in our example.

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

Theorem 1.3. *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 *states. Specifically, for any set* Π *and any oblivious sequence of reward functions, Algorithm* 3 *given in section* 2.3 *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 1.3 is sublinear, it is only slightly so. Unfortunately, this is unavoidable.

Theorem 1.4. *There is a set of* $k = 3$ *reference policies over* $S = 3$ *states and* $n = 3$ *actions with respect to which no player algorithm can guarantee expected regret better than* $O(T/\log^{3/2} T)$. *Moreover, this negative result holds in the commute example given in section* 1.1 *in which the reference policies are all reactive (as in Definition* 1.2*).*

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 1.5.

1.3. Discussion. A unifying paradigm for virtually all previous sequential optimization algorithms, whether in the expert or the bandit setting, is the following. As 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). For example, in the full-feedback analogue of our setting where the rewards of all actions are observed on each round, the player is able to "simulate" each of his contending policies and keep track of their cumulative rewards. Hence, he can treat each policy as an independent "expert" and use standard online learning techniques (such as the randomized weighted majority algorithm) to obtain $O(\sqrt{T})$ regret in this setting.

This typical learning paradigm is not suitable for our stateful policies model in conjunction 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 by 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.

To illustrate this complication, let us consider the standard black-box reduction from partial information to full-information models (see, e.g., Chapter 4 in [19]) and see why it fails in the context of stateful policies. The reduction is based on separate exploration and exploitation mechanisms: it dedicates certain rounds, chosen at random, for exploring the rewards of all actions, and uses the remaining rounds for exploiting actions based on past exploration. In particular, the procedure requires the player to switch repeatedly between different actions for incorporating exploration between exploitation rounds.

Notice the emerging difficulty in the context of stateful policies: whenever the player desires to switch from exploitation of one policy to exploration of another, he has no way of telling the current state of the new policy (which is required for producing its prediction) because he has been exploiting a different policy and, as a consequence, did not observe the past rewards required for determining the up-to-date state of the new policy. This complication in mixing-in exploration rounds is the major hurdle in using typical regret-minimization techniques for obtaining positive results in the stateful policies model.

1.4. Related work. 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 $\Theta(T/\text{polylog}(T))$. For this reason, we believe that the problem we consider is substantially different from previously studied, seemingly related sequential decision problems.

The full-feedback analogue of our setting is known to be captured by the so-called experts framework and has been studied under the name of "simulatable experts" [5]. As we mentioned earlier, standard online learning techniques yield $O(\sqrt{T})$ 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

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

between the two feedback models was observed in the multiarmed 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].

Among models with bandit feedback, the one most closely related to ours is perhaps the setting of the Exp4 algorithm [3], which is a variation on the standard multiarmed bandit problem. In this setting, on each round of the game, before committing to a single action and observing its reward the player is provided with the advice of a fixed set of "experts" on which arm to choose. The player's goal is to perform as well as the best expert in the set, and his regret is computed with respect to that expert. Auer et al. [3] suggest the Exp4 algorithm for this setup and prove that it achieves an optimal $O(\sqrt{T})$ bound over the regret. The crucial difference between this setting and ours is in the fact that the advice of an expert is assumed to be available at all times, whereas the advice of a stateful policy becomes unavailable once the player deviates from it. In other words, while our policies are simple algorithms that observe bandit feedback, we think of their experts as "oracles" whose observation is not limited to the player's rewards.

Our setting might seem reminiscent of (online) reinforcement learning models, and in particular, of online Markov decision processes (MDPs) [10, 18, 7]. In these models, there is typically a finite number of states, and the player's actions on each round cause him to transition from one state to another. As a consequence, the reward of the player on each round is determined not only based on his action on that round but also as a function of his actions in previous rounds. In contrast, in our setting the environment is oblivious and thus determines the reward based solely on the player's action on the current round. Furthermore, in an MDP the state is *of the environment* and the player's actions inevitably cause this state to change from round to round; in our model, the state is *owned by the player* (more precisely, by his contending policies) and he may freely transition himself to an arbitrary state (of any one of the policies) at any given moment, or even choose not to be in any of the states.

More generally, the settings considered in the related works [16, 11, 1] (among others), which deal with stateful and reactive environments in an online decision making framework, are also substantially different from ours. As is the case with the reinforcement learning literature, the focus of these works is the adaptiveness of the adversary and not of the player's reference policies.

Finally, we remark that our definition of a stateful policy is not new and similar notions have been considered in the past. Most notably, the work of Feder, Merhav, and Gutman [12] in the related context of binary sequence prediction considers a similar concept which they call the "FS predictor" and studies the prediction power of the class of all such predictors with at most S states. However, our goal is entirely different from theirs: while they are concerned with the prediction power of the class of all such predictors with at most S states, we aim to understand the difficulty of learning a small set of these concepts (with bandit feedback).

1.5. 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 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 distinguish between two different modes a player in the stateful policies model may be in, at any given round: the

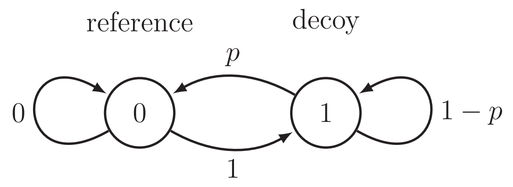


FIG. 2. *The dynamics of the* switch *action in the hidden bandit model can be viewed as a two-state Markov chain, where state* 0 *stands for the reference arm and state* 1 *for the decoy arm. The arrow labels denote the corresponding arm transition probabilities.*

"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 (following the best reference policy in an incorrect state, following other reference policies, or executing a sequence of actions that do not correspond to any reference policy). The player might not be 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 onward 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 multiarmed 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 multiarmed 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, illustrated in Figure 2. 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 total reward of the *reference arm*, namely,

$$\text{Regret}_T \;=\; \sum_{t=1}^{T} r_t(0) - \mathbf{E}\left[\sum_{t=1}^{T} r_t(X_t)\right] ,$$

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.

This completes the description of the hidden bandit problem.

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, which we later show how to translate into the corresponding upper and lower bounds in the stateful policies model.

THEOREM 1.5. *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 2.2 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 1.6. *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 1.5 and the lower bound of Theorem 1.6 that translates into a gap between our main upper and lower bounds of Theorems 1.3 and 1.4. 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 section 2.6.)

THEOREM 1.7. *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 1.7: for the positive results a *Markovian* algorithm suffices, and for the negative results a *constant* adversary suffices. See section 2.6 for more details.

1.6. Our techniques and additional related work. Our algorithm in the proofs of Theorems 1.5 and 1.3 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 2.1 (see Lemma 2.4).

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 1.5.)

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 the 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 2.4 (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 (almost in the sense that 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 at a random scale a sequence of values has small Fourier coefficients, and small Fourier coefficients correspond to not having much variability at this scale. Indeed, a variant of the local repetition lemma in the special case of $d = 2$ is reproved in [13] using Parseval's identity for the so-called Haar transform. Moreover, it turns out that the exact same special case was also handled (while proving nearly the same bounds) as part of earlier work of Drucker on certain prediction tasks [8].

Our lower bound of Theorem 1.6 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 [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 shares similarities with a construction used by Dwork et al. [9] 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 1.7 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 1.7 (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 [20] on monotonicity testing with a small number of queries.

1.7. Extensions of our upper bound. We discuss a few simple extensions of the basic model presented in section 1.1.

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 1.4) 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 1.3) also holds without change when reference policies are time dependent—nothing in the proof of Theorem 1.3 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 1.4) 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 1.3) 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 1.3 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" [17] and "bounded memory adaptive adversary" [1]. See section 2.7 for more details.

2. Proofs.

2.1. 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 1.5. 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 a 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 is 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 2.1 (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 2.2.

Definition 2.2 (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 d-sampling process from a string s can be alternatively described as sampling a node from a d-ary tree with k levels whose leaves correspond to the $n = d^k$ characters of s, as follows: first, a level $0 \leq \ell < k$ is chosen uniformly at random, and then a node is picked uniformly at random from the nodes in that level; the output of the process is the substring formed by the characters corresponding to the leaves of the subtree whose root is the sampled node (which is necessarily not a leaf by itself). See also Figure 3 for a visual example of this process.

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 2.1.

Remark. In Definition 2.2 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 2.2. 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 2.2 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 2.3 (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 2.2) is (d,ϵ)-repetitive (as in Definition 2.1).*

The main result of this section is the following.

Lemma 2.4 (local repetition lemma). *Let d be a positive integer, and $\epsilon, \delta > 0$. Then for every $n > d^k$ where $k = d/(4\epsilon^2\delta)$, every string $s \in [0,1]^n$ is (d,ϵ,δ)-locally repetitive.*

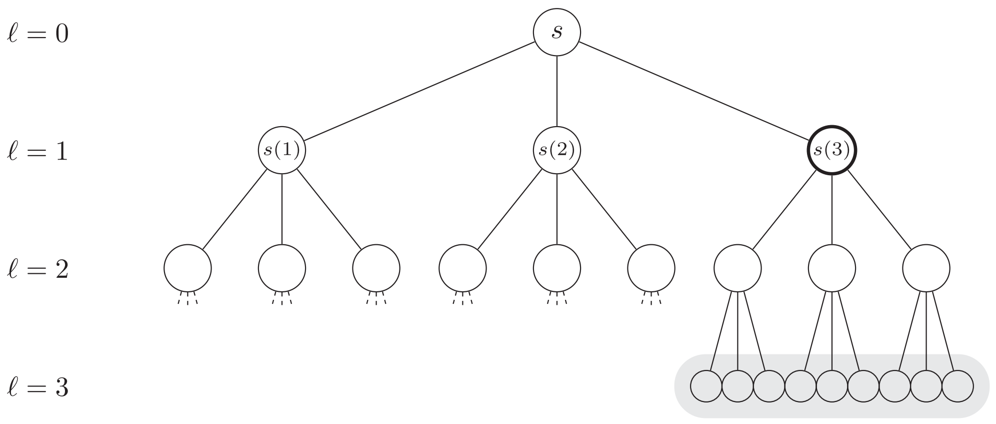


Fig. 3. *An illustration of the d-sampling process with $d = 3$ and $k = 3$. First, a level of the d-ary tree is picked uniformly at random among all levels but the last, say, level $\ell = 1$; then, a node within that level is chosen uniformly at random, say, the rightmost one (marked in black). The output of d-sampling in this case is the substring $s(3)$ formed by the characters that correspond to the descendent leaves of the sampled node, marked in gray.*

Proof. For simplicity, we shall assume that $1/\epsilon$, $1/\delta$, and k 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 2.2 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 ℓ to be

$$\forall\, 0 \le \ell \le k\ , \qquad V_\ell \;=\; \frac{1}{d^\ell} \sum_{v \,\in\, \ell} (x_v)^2\ .$$

Proposition 2.5. *With the above definition, we have $V_k - V_0 \le \frac{1}{4}$.*

Proof. By definition, $V_0 = (x_s)^2$. On the other hand, $V_k \le \frac{1}{d^\ell} \sum_{v \,\in\, \ell} x_v = x_s$ since $0 \le x_v \le 1$ for all substrings v. Hence, $V_k - V_0 \le x_s - (x_s)^2$ and the difference on the right-hand side is maximized when $x_s = \frac{1}{2}$, giving a value of $\frac{1}{4}$. ∎

The variability 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 2.6. *If v is an aligned string that is not (d,ϵ)-repetitive, then*

$$\frac{1}{d} \sum_{i=1}^{d} (x_{v(i)})^2 \;>\; (x_v)^2 + \frac{\epsilon^2}{d}\ .$$

Proof. If v is not (d,ϵ)-repetitive, then it has at least one substring $v(i)$, with $|x_v - x_{v(i)}| > \epsilon$. Hence $\sum_{i=1}^d (x_v - x_{v(i)})^2 > \epsilon^2$. By definition, $x_v = \frac{1}{d}\sum_{i=1}^d x_{v(i)}$. Hence $\sum_{i=1}^d (x_v - x_{v(i)})^2 = \sum_{i=1}^d (x_{v(i)})^2 - d(x_v)^2$. Putting these two facts together we get that $\sum_{i=1}^d (x_{v(i)})^2 > d(x_v)^2 + \epsilon^2$, implying the proposition. ∎

We now turn to prove the lemma. Let v be a random substring obtained as a result of d-sampling. Define δ_ℓ to 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 2.6 level by level implies that

$$\begin{aligned} V_k &= \frac{1}{d^k}\sum_{v\,\in\,k}(x_v)^2 \\ &> \frac{1}{d^{k-1}}\sum_{v\,\in\,k-1}(x_v)^2 + \frac{\epsilon^2}{d}\delta_{k-1} \;>\; \cdots \\ &> (x_s)^2 + \frac{\epsilon^2}{d}\sum_{\ell=0}^{k-1}\delta_\ell \\ &= V_0 + \frac{k\delta\epsilon^2}{d}\;. \end{aligned}$$

Contrasting this with Proposition 2.5 we obtain $\frac{k\delta\epsilon^2}{d} < \frac{1}{4}$, implying that $\delta < \frac{d}{4\epsilon^2 k}$. ☐

In Appendix A we provide an alternative proof for Lemma 2.4. Though that proof gives somewhat weaker bounds, we find it informative, as it shows the relation between the lemma and martingale theory.

Lemma 2.4 is best possible in the following sense.

LEMMA 2.7. *There is a universal constant $c > 0$ such that the following holds. Let d be a positive integer, and $0 < \epsilon, \delta < \frac{1}{2}$. Then there exists a string $s \in [0,1]^n$, where $n > d^k$ with $k = cd/(\epsilon^2\delta)$, that is not (d,ϵ,δ)-locally repetitive.*

Proof. Again, we assume for simplicity that $1/\epsilon$ and $1/\delta$ are integers. Fix d, k and let $n = d^k$. Given $\epsilon > 0$, pick $\eta > \epsilon/2$ to be as small as possible, conditioned on $1/2\eta$ being an integer. Construct a string $s \in [0,1]^n$ in a top-down manner, by associating the x_v variables with the possible choices of substrings v in the d-sampling scheme. Start by setting $x_s = \frac{1}{2}$. Thereafter, for every substring v for which x_v is already determined do the following. If v is a single character, nothing needs to be done. Else, v represents a string whose length is a multiple of d. Let $v(0), \ldots, v(d-1)$ denote the d substrings whose concatenation gives v. If either $x_v = 0$ or $x_v = 1$, then for every i, let $x_{v(i)} = x_v$. In this case v is (d,ϵ)-repetitive. However, in every other case, let $x_{v(0)} = x_v + \eta$, $x_{v(1)} = x_v - \eta$, and $x_{v(m)} = x_v$ for all $2 \le m \le d-1$. In this case v is not (d,ϵ)-repetitive. Figure 4 shows a possible instance of this construction.

The construction above maintains that $0 \le x_v \le 1$ for every v, and moreover, $x_v = \frac{1}{d}\sum_{i=1}^{d} x_{v(i)}$. Hence the x_v variables indeed represent the true averages over the corresponding substrings of s.

For an individual character at location i in the string s, its value is integer (0 or 1) if and only if when writing i in base d (namely, $i = a_{k-1}d^{k-1} + \ldots a_1 d + a_0$, with $0 \le a_j \le d-1$), there is a value $0 \le j \le k-1$ such that among the coefficients $a_j, \ldots, a_{k-1}$, there is a difference of at least $1/2\eta$ between the number of those coefficients that are 0 and the number of those coefficients that are 1; see Figure 4 for a demonstration of this argument.

Let us now set $k = cd/\eta^2$ for some small universal constant $c > 0$ (independent of d and η). It is not difficult to argue that in this case, only a small fraction of the characters of s are integers (i.e., either 0 or 1). (For a random i, only $O(k/d)$ of its digits in base d are 0/1, and for most sequences of ± 1 of length m, there is no prefix whose sum exceeds $O(\sqrt{m})$ in absolute value.) Hence for this value of k, almost no v is (d,ϵ)-repetitive.

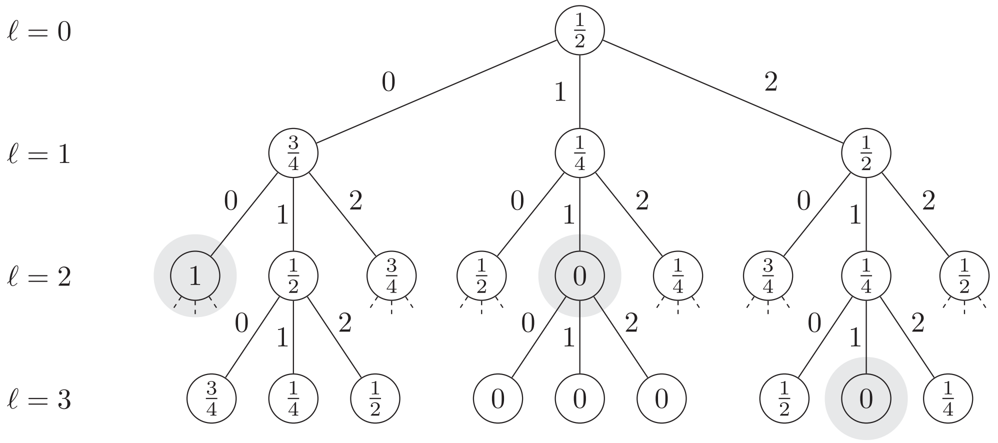


FIG. 4. *An example of the construction in Lemma* 2.7 *with $d = 3$, $k = 3$, and $\eta = 1/4$. The digits along the path leading to each leaf form the d-ary representation of the corresponding character's index in the string. A leaf has a value of* 0 *(respectively,* 1*) if and only if the path leading to one of its ancestors (possibly the leaf itself) has two more* 1*'s than* 0*'s (respectively, two more* 0*'s than* 1*'s) along its edges. Three examples of ancestors that meet this condition are marked in gray.*

Finally, set $k = k_0/2\delta$, with $k_0 = cd/\eta^2$. With probability 2δ the d-sampling procedure will choose a level among the top k_0 levels, and then with probability at least $\frac{1}{2}$ the sampled string v will not be (d,ϵ)-repetitive. ∎

2.2. 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 2.3) 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 2.1).

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 algorithm 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.

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.

LEMMA 2.8. *Assume that the reward sequence of the reference arm is (d,ϵ)-repetitive, with $d \geq (1/p^2\epsilon)\log^2(1/\epsilon)$. Then the expected regret of Algorithm* 1 *is at most $8\epsilon T$.*

Proof. Let v denote the average reward (over all rounds $t = 1,2,\ldots,T$) of the reference arm. Notice that the probability of not visiting this arm in the first phase of the algorithm is no more than $(1-p)^m \leq e^{-pm} \leq \epsilon$. Hence, with probability at least $1-\epsilon$, the first phase of the algorithm samples the reference arm at least once,

```
• let $m = \lceil (1/p)\log(1/\epsilon) \rceil$
• **Phase I:** for $i = 1, \dots, m$: 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 $\bar{r}_1, \dots, \bar{r}_m$ in descending order to obtain $\bar{r}_1 \geq \dots \geq \bar{r}_m$
• **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: An algorithm for (d, ϵ)-repetitive reference sequences (parameters: d, ϵ, p, T).

so that there exists some $j \in [m]$ for which $\bar{r}_j \in [v - \epsilon, v + \epsilon]$ as the reward sequence of the reference arm is assumed to be (d, ϵ)-repetitive. The total regret incurred in this phase is bounded by $m \cdot T/d$.

Next, assume that indeed $\bar{r}_j \in [v - \epsilon, v + \epsilon]$ for some $j \in [m]$ and consider the second phase of the algorithm. Notice that once $i = j$ and the reference arm is selected, the algorithm stops switching and stays on that arm until the game ends. This is true because the reference arm is (d, ϵ)-repetitive, so each average reward $\bar{r}$ encountered when this arm is selected exceeds $v - \epsilon \geq \bar{r}_j - 2\epsilon$.

Let us bound the regret incurred in the second phase of the algorithm until this event occurs (if at all). To this end, consider iterations with $i \leq j$. On any such iteration in which $\bar{r} \geq \bar{r}_i - 2\epsilon$, we have $\bar{r} \geq \bar{r}_j - 2\epsilon \geq v - 3\epsilon$ so that the incurred regret is at most $4\epsilon \cdot T/d$. The number of iterations that fail to satisfy this condition is equal to the number of switch actions issued by the algorithm. The number of switch actions in iterations with $i < j$ is no more than $(j-1) \cdot m$, and once $i = j$, the algorithm hits the reference arm after at most m switch actions with probability $1 - \epsilon$ (and subsequently stops switching). Thus, with probability at least $1 - \epsilon$ the total number of switch actions is bounded by $(j-1) \cdot m + m \leq m^2$. In this case, the total regret incurred in the second phase is at most $4\epsilon T + m^2 \cdot T/d$. Overall, the total regret in both phases is then bounded by

$$4\epsilon T + \frac{T}{d} \cdot m^2 + \frac{T}{d} \cdot m \;\leq\; 4\epsilon T + \frac{2T}{d} \cdot m^2 \;\leq\; 4\epsilon T + \frac{2T}{d} \cdot \frac{2}{p^2} \log^2 \frac{1}{\epsilon} \;\leq\; 8\epsilon T \ ,$$

where we have used our assumption that $d \geq (1/p^2\epsilon)\log^2(1/\epsilon)$.

On the other hand, if one of the phases fails, then the total regret might be as large as T, but this happens with probability at most 2ϵ. Hence, the expected regret of the algorithm is at most $10\epsilon T$. ∎

Our general algorithm, which 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.

We are now ready to give the main result of this section, which gives an upper bound over the expected regret of Algorithm 2.

Theorem 1.5 (restated). *The expected regret of Algorithm 2 is*

$$O\left(\frac{1}{\sqrt{p}} \cdot \frac{T \log\log T}{\log^{1/4} T}\right) \ .$$

The proof uses the local repetition lemma, restated here for convenience.

```
• set
      $\epsilon = \frac{1}{\sqrt{p}} \cdot \frac{\log\log T}{\log^{1/4} T}$ ,    $d = \left\lceil \frac{1}{p^2\epsilon} \log^2 \frac{1}{\epsilon} \right\rceil$
• 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 $d, \epsilon, p, b$
```

Algorithm 2: An algorithm for the hidden bandit problem that guarantees sublinear expected regret (parameters: p, T).

Lemma 2.4 (restated). *Let d be a positive integer and $\epsilon, \delta > 0$. Then for every $n > d^k$ where $k = d/(4\epsilon^2\delta)$, every string $s \in [0,1]^n$ is (d,ϵ,δ)-locally repetitive.*

Proof of Theorem 1.5. Set $d = (1/p^2\epsilon)\log^2(1/\epsilon)$, $\delta = \epsilon$, $k = d/4\epsilon^3$ in Lemma 2.4 (for simplicity, we assume that d and k are integers), which then states that any sequence of length at least $T_\epsilon = d^k$ is (d,ϵ,ϵ)-locally repetitive. Notice that for $T \geq \Omega(1)$ and for our choice of ϵ we have

$$\begin{aligned}
\log T_\epsilon &= \frac{1}{4p^2\epsilon^4}\log^2\left(\frac{1}{\epsilon}\right)\cdot\log\left(\frac{1}{p^2\epsilon}\log^2\frac{1}{\epsilon}\right)\\
&\leq \frac{1}{p^2\epsilon^4}\log^4\left(\frac{1}{p^2\epsilon^4}\right) = \frac{\log T}{\log^4(\log T)}\cdot\log^4\left(\frac{\log T}{\log^4(\log T)}\right)\\
&\leq \log T;
\end{aligned}$$

thus $T_\epsilon \leq T$, 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 $d = (1/p^2\epsilon)\log^2(1/\epsilon)$. Hence, according to Lemma 2.8, 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. ∎

2.3. 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, which 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.

We now prove our main upper bound result, which provides a regret guarantee for Algorithm 3.

Theorem 1.3 (restated). *For any reference set Π of k stateful policies over S states, the expected regret of Algorithm 3 with respect to Π is*

$$O\left(\sqrt{kS}\cdot\frac{T\log\log T}{\log^{1/4} T}\right).$$

Proof. 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

```
• 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 the action $f^{\pi_t}(s_t)$, observe reward $r$, and update $\pi_{t+1} \leftarrow \pi_t$ and $s_{t+1} \leftarrow g^{\pi_t}(s_t, r)$
    – 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: An algorithm for competing with stateful policies (parameters: Π, T).

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

$$\forall\, t \qquad X_t \;=\; \mathbb{1}_{\pi_t \neq \pi^* \,\vee\, s_t \neq s_t^*} \;.$$

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 1.5 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 toward 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 $1/kS$ 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 k and save a $k^{1/4}$ factor in the resulting bound, albeit with a more involved algorithm.

2.4. Lower bound for hidden bandits. In this section we prove our lower bound for the hidden bandit problem with $p = 1/2$ given in Theorem 1.6, which we restate here more formally.

THEOREM 1.6 (restated). *For any randomized player strategy in the hidden bandit model with $p = 1/2$, there exists an oblivious sequence of reward functions $r_1, \ldots, r_T$ that forces the player to incur an expected regret of $\Omega(T/\log^{3/2} T)$ with respect to the reference arm.*

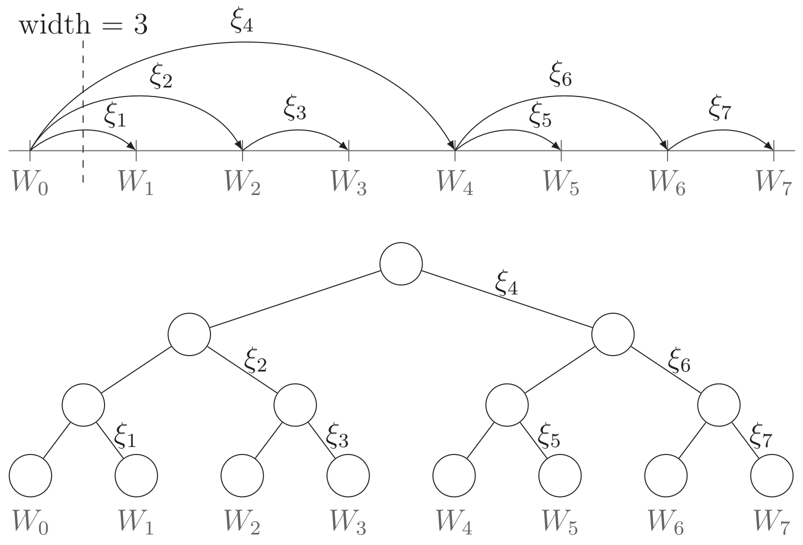


FIG. 5. *(Taken from* [6]*.) An illustration of the MRW random process with $T = 7$. (Top) A dependency graph of the random walk, which for each time t includes an incoming directed edge from its parent $\rho(t)$ in the process. The maximal cut of this walk (see Definition* 2.12*), whose size is* 3*, is achieved at $t = 1$; hence, the width of the walk is* 3*. (Bottom) The random walk can be equivalently described as the sequence of random variables at the leaves of a binary tree obtained by summing the variables ξ_t's on the right edges along the corresponding paths from the root.*

In order to prove Theorem 1.6 we make use of Yao's principle [21], 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 1.6 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 multiscale random walk (MRW) stochastic process of Dekel et al. [6].

DEFINITION 2.9 (MRW [6]).

Given a sequence $\xi_1, \dots, \xi_T$ of independent and identically distributed (i.i.d.) random variables, the MRW process $W_{0:T}$ is defined recursively by

$$\begin{aligned} & & W_0 &= 0 \ , \\ (1) \qquad & \forall\, t \in [T] & W_t &= W_{\rho(t)} + \xi_t \ , \end{aligned}$$

where

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

Figure 5 gives an illustration of the MRW process for $T = 7$ and shows how each variable in the random walk is obtained from the preceding variables in the walk. An alternative, equivalent formulation of the same random process is demonstrated in the bottom part of the figure: one can construct a binary tree with T leaves corresponding to the rounds of the game, in which the sum of the ξ_t variables along the path from the root to a leaf yields the walk variable associated with that round. For further details on the MRW stochastic process and its properties, refer to [6].

Strategy for the hidden bandit adversary (parameters: T)

- set

$$\epsilon = \frac{1}{320 \log_2^{3/2} T} \qquad \text{and} \qquad \gamma = \frac{1}{4 \log_2 T}; \tag{2}$$

- define $W_{1:T}$ to be an MRW process generated according to (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

$$\begin{aligned} \forall\, t \in [T] \qquad \widetilde{\Gamma}_t(x) &= \tfrac{1}{2} + W_t - \epsilon\, \mathbb{1}_{x \neq 0}\ , \\ \Gamma_t(x) &= \operatorname{clip}\big(\widetilde{\Gamma}_t(x)\big)\ , \end{aligned}$$

  where $\operatorname{clip}(r) = \min\{\max\{r,0\},1\}$.

FIG. 6. *An oblivious strategy for the adversary that forces a regret of $\Omega(T/\log^{3/2} T)$ for any algorithm for the hidden bandit problem with $p = 1/2$.*

Our construction, described in Figure 6, is similar to the one used by Dekel at 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 analogue 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.

For this construction, we prove the following lower bound on the performance of any deterministic algorithm that immediately implies Theorem 1.6.

THEOREM 2.10. *The expected regret of any deterministic player algorithm on the stochastic sequence of reward functions $\Gamma_{1:T}$ defined in Figure 6 is at least $10^{-4} \cdot T/\log_2^{3/2} T$.*

Before we begin with the analysis, we recall two key combinatorial properties of the MRW process that are essential to our analysis. See [6] for more details and the formal proofs.

DEFINITION 2.11 (depth). *Given a parent function ρ, the set of ancestors of t is denoted by $\rho^*(t)$ and defined as the set of positive indices that are encountered when ρ is applied recursively to t. Formally, $\rho^*(t)$ is defined recursively as*

$$\begin{aligned} \rho^*(0) &= \{\}, \\ \forall\, t \in [T] \qquad \rho^*(t) &= \rho^*\big(\rho(t)\big)\ \cup\ \{\rho(t)\}\ . \end{aligned} \tag{4}$$

The depth *of ρ is then defined as $d(\rho) = \max_{t \in [T]} |\rho^*(t)|$.*

DEFINITION 2.12 (cut, width). *Given a parent function ρ, define*

$$cut(t) = \{s \in [T]\ :\ \rho(s) < t \leq s\}\ ,$$

the set of rounds that are separated from their parent by t. The width *of ρ is then defined as $w(\rho) = \max_{t \in [T]} |\text{cut}(t)|$.*

LEMMA 2.13. *The depth and width of the MRW are both upper-bounded by* $\lfloor \log_2 T \rfloor + 1$.

We begin the analysis with some notation. Fix some deterministic player strategy that generates a sequence of random variables $X_{1:T}$ when faced with the random reward functions $\Gamma_{1:T}$, where $X_t \in \{0,1\}$ is the arm pulled on round t of the game. We let $Y_t = \widetilde{\Gamma}(X_t)$ denote the unclipped reward encountered on round t by following the strategy (which is not directly observable to the player).

The strategy induces a partition of the rounds into *epochs*, where epoch m spans over rounds between the player's $m-1$ and mth switch actions. Let $\chi_m \in \{0,1\}$ denote the arm pulled throughout epoch m, and let T_m denote the length of that epoch. We set $T_m = 0$ if the mth epoch does not take place (that is, if the player makes less than $m-1$ switch actions throughout the game). Without loss of generality, we may assume that there are exactly T epochs corresponding to $m = 1, 2, \ldots, T$, some of which are of zero length.

Without loss of generality, we may assume that the assignment of arms to epochs is determined before the game begins. Namely, a sequence of random variables $\chi_1, \chi_2, \ldots, \chi_T$ is drawn ahead of time from the distribution described by the Markov chain, where $\chi_m \in \{0,1\}$ is the index of the arm assigned to the player on the mth epoch (some of these variables may eventually not be used). Notice that as we assume the player to be deterministic, the set of random variables $\xi_{1:T}$ and $\chi_{1:T}$ completely determine the outcome of the game.

A key step in our analysis is to show that even if we allow the player to observe the entire sequence $Y_{1:T}$ directly, he is unable to detect (with sufficient confidence) an epoch during which the reference arm was pulled. To this end, for each epoch $m \in [T]$ we define two conditional probability measures,

$$\begin{aligned} P_m(\cdot) &= \Pr(\,\cdot \mid \chi_m = 0), \\ Q_m(\cdot) &= \Pr(\,\cdot \mid \chi_m = 1), \end{aligned}$$

over the sigma algebra $\mathcal{F} = \sigma(Y_{1:T})$ generated by the variables $Y_{1:T}$. Our first lemma shows that any event observable to the player (i.e., one that relies on the random variables the player receives as feedback) which is likely to occur assuming $\chi_m = 0$ is also likely to occur given $\chi_m = 1$.

LEMMA 2.14. *For all epochs m and for any event $A \in \mathcal{F}$ it holds that* $Q_m(A) \geq \frac{1}{4e} \cdot P_m(A)$.

Proof. Fix some epoch $m \in [T]$. In order to prove the lemma, we bound the log-likelihood ratio of an arbitrary feasible realization $y_{1:T}$ of the variables $Y_{1:T}$ between the measures P_m and Q_m. To do that, we condition on the variables $\chi_1, \ldots, \chi_{m-1}, \chi_{m+1}, \ldots, \chi_T$ corresponding to the arms pulled in other epochs. Consider two realizations $x_{1:T}, x'_{1:T} \in \{0,1\}^T$ of the sequence $\chi_{1:T}$ that differ only by the value assigned to the mth variable, with $x_m = 0$ and $x'_m = 1$, and define the measures

$$\begin{aligned} P'_m(\cdot) &= P_m(\,\cdot \mid \chi_{1:T} = x_{1:T})\,, \\ Q'_m(\cdot) &= Q_m(\,\cdot \mid \chi_{1:T} = x'_{1:T})\,. \end{aligned}$$

Then, we can write

$$\begin{aligned} \log \frac{P_m(Y_{1:T} = y_{1:T},\, \chi_{1:T} = x_{1:T})}{Q_m(Y_{1:T} = y_{1:T},\, \chi_{1:T} = x'_{1:T})} &= \log \frac{P_m(\chi_{1:T} = x_{1:T})}{Q_m(\chi_{1:T} = x'_{1:T})} \\ &+ \log \frac{P'_m(Y_{1:T} = y_{1:T})}{Q'_m(Y_{1:T} = y_{1:T})}\,. \end{aligned} \tag{5}$$

In order to bound the first term on the right-hand side, note that the Markov-chain dynamics of the switch action (recall Figure 2) imply

$$\frac{P_m(\chi_{m+1} = x_{m+1})}{Q_m(\chi_{m+1} = x_{m+1})} = \frac{\Pr(\chi_{m+1} = x_{m+1} \mid \chi_m = 0)}{\Pr(\chi_{m+1} = x_{m+1} \mid \chi_m = 1)} \le \frac{1}{1/2} = 2 .$$

Similarly, using Bayes' law,

$$\begin{aligned}\frac{P_m(\chi_{m-1} = x_{m-1})}{Q_m(\chi_{m-1} = x_{m-1})} &= \frac{\Pr(\chi_{m-1} = x_{m-1} \mid \chi_m = 0)}{\Pr(\chi_{m-1} = x_{m-1} \mid \chi_m = 1)} \\ &= \frac{\Pr(\chi_m = 0 \mid \chi_{m-1} = x_{m-1})}{\Pr(\chi_m = 1 \mid \chi_{m-1} = x_{m-1})} \cdot \frac{\Pr(\chi_m = 1)}{\Pr(\chi_m = 0)} \\ &\le \frac{1/2}{1/2} \cdot \frac{2/3}{1/3} = 2 ,\end{aligned}$$

where we have used the fact that the Markov chain is in its stationary distribution $(\frac{1}{3}, \frac{2}{3})$. Hence, recalling that the sequences $x_{1:T}$ and $x'_{1:T}$ only differ by their mth element and using the Markov property of $\chi_{1:T}$, we conclude

$$\begin{aligned}\log \frac{P_m(\chi_{1:T} = x_{1:T})}{Q_m(\chi_{1:T} = x'_{1:T})} &= \log \frac{P_m(\chi_{m-1} = x_{m-1})}{Q_m(\chi_{m-1} = x_{m-1})} + \log \frac{P_m(\chi_{m+1} = x_{m+1})}{Q_m(\chi_{m+1} = x_{m+1})} \\ &\le 2 \log 2 . \qquad (6)\end{aligned}$$

For bounding the second term on the right-hand side of (5), we decompose it into a sum using the fact that Y_t is conditionally independent of all Y_s with $s \ne \rho(t)$ given $Y_{\rho(t)}$ under both P'_m and Q'_m (recall (1)), as follows:

$$\log \frac{P'_m(Y_{1:T} = y_{1:T})}{Q'_m(Y_{1:T} = y_{1:T})} = \sum_{t=1}^{T} \log \frac{P'_m(Y_t = y_t \mid Y_{\rho(t)} = y_{\rho(t)})}{Q'_m(Y_t = y_t \mid Y_{\rho(t)} = y_{\rho(t)})} .$$

Here, for convenience, we define a fictitious deterministic reward $Y_0 = y_0 = \frac{1}{2}$. Each term in the above sum corresponds to an edge in the dependency graph of the MRW process, formed by the function ρ. Consider a particular term of the sum that represents the edge $(\rho(t), t)$, and let i and j denote the epochs containing the end points $\rho(t)$ and t, respectively. Notice that the conditional distribution of Y_t given $Y_{\rho(t)}$ is determined only by the values of χ_i and χ_j. In particular, if $\chi_i = \chi_j$, then $Y_t = Y_{\rho(t)} + \xi_t$. However, if $\chi_i = 0,\ \chi_j = 1$, then $Y_t = Y_{\rho(t)} + \xi_t - \epsilon$, and if $\chi_i = 1,\ \chi_j = 0$, then $Y_t = Y_{\rho(t)} + \xi_t + \epsilon$. Hence, the log-likelihood term is zero unless $Y_t \mid Y_{\rho(t)}$ has different distributions under the measures P'_m and Q'_m, which can happen only when either $i = m$ or $j = m$ (but not both) since the realizations $x_{1:T}, x'_{1:T}$ differ only in the value assigned to χ_m. In the latter case, the log-likelihood term is equal to the log-likelihood ratio between the distributions of ξ_t and $\xi_t \pm \epsilon$ at some point in their (common) support, which can be at most γ, as can be seen from (3).

Now, notice that any edge $(\rho(t), t)$ for which either $i = m$ or $j = m$ is in $\mathrm{cut}(S_m)$ or in $\mathrm{cut}(S_{m+1})$, where S_m and S_{m+1} denote the rounds on which epochs m and $m+1$ begin. Overall, we get

$$\log \frac{P'_m(Y_{1:T} = y_{1:T})}{Q'_m(Y_{1:T} = y_{1:T})} \le \gamma \cdot |\mathrm{cut}(S_{m-1})| + \gamma \cdot |\mathrm{cut}(S_m)| \le 2\gamma\, w(\rho) \le 1 ,$$

where we have used the fact that the width of the MRW process is upper-bounded by $2\log_2 T$ and our choice of γ in (2). Plugging this and (6) into (5) and exponentiating results in

$$P_m(Y_{1:T} = y_{1:T},\, \chi_{1:T} = x_{1:T}) \;\geq\; \frac{1}{4e} \cdot Q_m(Y_{1:T} = y_{1:T},\, \chi_{1:T} = x'_{1:T}) \;.$$

Since this holds for any assignment of the variables χ_i $(i \neq m)$, by marginalizing over these variables we obtain that $P_m(Y_{1:T} = y_{1:T}) \geq \frac{1}{4e} \cdot Q_m(Y_{1:T} = y_{1:T})$ for any sequence $y_{1:T}$. Finally, integrating this inequality over the event $A \in \mathcal{F}$ gives the lemma. ◻

We now turn back to analyzing the player's regret. In order to lower-bound the expected regret of the player's action sequence $X_{1:T}$, it will be convenient for us to first analyze the regret of the *same sequence* as measured by the unclipped reward functions $\widetilde{\Gamma}_t$, namely,

$$\widetilde{R} \;=\; \sum_{t=1}^{T} \widetilde{\Gamma}_t(0) - \sum_{t=1}^{T} \widetilde{\Gamma}_t(X_t) \;,$$

and later deal with the effect of the clipping. This quantity can be alternatively expressed as a simple function of the variables T_m and χ_m,

$$\widetilde{R} \;=\; \sum_{m=1}^{T} \widetilde{R}_m \;, \qquad \text{where} \qquad \forall\, m \quad \widetilde{R}_m \;=\; \epsilon T_m \cdot \mathbb{1}_{\chi_m \neq 0} \;.$$

Here, $\widetilde{R}_m$ is the regret incurred during epoch m (in terms of the functions $\widetilde{\Gamma}_{1:T}$) and $\widetilde{R}$ is simply the total regret incurred in all epochs. The following lemma relates the quantity $\mathbf{E}[\widetilde{R}]$ to the actual expected regret of the player.

LEMMA 2.15. *The player's expected regret can be lower-bounded as* $\mathbf{E}[R] \geq \mathbf{E}[\widetilde{R}] - \epsilon T/25$.

Proof. We first prove that for each $t \in [T]$, with probability at least $1 - 1/25$, both of the rewards $\widetilde{\Gamma}_t(0), \widetilde{\Gamma}_t(1)$ lie in the interval $[0, 1]$. Then, the lemma would follow since this proves that in expectation there are at most $T/25$ rounds on which the reward functions $\Gamma_t(x)$ and $\widetilde{\Gamma}_t(x)$ do not coincide. On each of those rounds, the player's regret might decrease by at most ϵ since the gap between the arms is only narrowed by the clipping of the rewards.

To prove the above claim, fix some $t \in [T]$. Since the depth of the process $W_{1:T}$ is bounded by $2\log_2 T$ (see Lemma 2.13), the random variable in W_t is a sum of at most $2\log_2 T$ variables of the sequence $\xi_{1:T}$. For bounding the magnitude of each of the variables ξ_s, which are distributed according to (3), let us first bound their variance. Noticing that $|\xi_s|/\epsilon$ is a geometric random variable with parameter $p = 1 - e^{-\gamma}$ and using a standard bound of $2/p^2$ over the second moment of the geometric distribution gives

$$\mathrm{Var}(\xi_s) \;=\; \mathbf{E}[|\xi_s|^2] \;\leq\; \frac{2\epsilon^2}{(1 - e^{-\gamma})^2} \;\leq\; \frac{8\epsilon^2}{\gamma^2} \;,$$

where in the last inequality we have used the fact that $e^{-x} \leq 1 - x/2$ for $x \in [0, 1]$. Since the ξ_s's are independent, this implies that $\mathrm{Var}(W_t) \leq 16(\epsilon/\gamma)^2 \log_2 T$. By Chebyshev's inequality we now obtain $\Pr\left(W_t \geq 20(\epsilon/\gamma)\sqrt{\log_2 T}\right) \leq 1/25$, so that with probability at least $1 - 1/25$ we have

$$W_t \; < \; \frac{20\epsilon}{\gamma}\sqrt{\log_2 T} \; \le \; \frac{1}{4}$$

for our choice of ϵ and γ stated in (2).

Finally, recall that either $\widetilde{\Gamma}_t(x) = \frac{1}{2} + W_t$ or $\widetilde{\Gamma}_t(x) = \frac{1}{2} + W_t - \epsilon$, depending on whether $x = 0$. In any case, we have $\widetilde{\Gamma}_t(x) \in [0, 1]$ for all $x \in \{0, 1\}$ with probability at least $1 - 1/25$, since $\epsilon < 1/4$. ∎

Next, we use Lemma 2.14 to show that in expectation, the regret $\widetilde{R}_m$ incurred on epoch m grows linearly with the length of the epoch.

LEMMA 2.16. *For each epoch m we have $\mathbf{E}[\widetilde{R}_m \mid T_m = t] \ge \epsilon t/12$ for all t.*

Proof. Fix some $t \in \{0, 1, \ldots, T\}$, and notice that $\{T_m = t\} \in \mathcal{F}$ as the random variable T_m is observable to the player and, in particular, is a deterministic function of $Y_{1:T}$. Then

$$\mathbf{E}[\widetilde{R}_m \mid T_m = t] \; = \; \mathbf{E}[\epsilon T_m \cdot \mathbb{1}_{\chi_m \neq 0} \mid T_m = t] \; = \; \epsilon t \cdot \Pr(\chi_m \neq 0 \mid T_m = t) \; ,$$

and by Bayes' law, we have

$$\begin{aligned} \mathbf{E}[\widetilde{R}_m \mid T_m = t] \; &= \; \epsilon t \cdot \frac{\Pr(T_m = t \mid \chi_m \neq 0) \cdot \Pr(\chi_m \neq 0)}{\Pr(T_m = t)} \\ &\ge \; \frac{\epsilon t}{2} \cdot \frac{Q_m(T_m = t)}{\Pr(T_m = t)} \; . \end{aligned} \tag{7}$$

On the other hand, using Lemma 2.14 we obtain $P_m(T_m = t) \le 4e \cdot Q_m(T_m = t)$, which together with $\Pr(T_m = t) \; = \; \frac{1}{2}P_m(T_m = t) + \frac{1}{2}Q_m(T_m = t)$ gives

$$Q_m(T_m = t) \; \ge \; \frac{2}{1 + 4e} \cdot \Pr(T_m = t) \; \ge \; \frac{1}{6}\Pr(T_m = t) \; .$$

Plugging this into (7) concludes the proof. ∎

Theorem 2.10 is now a direct consequence of Lemmas 2.15 and 2.16.

Proof of Theorem 2.10. Applying Lemma 2.16, we can lower-bound the expected value of $\widetilde{R}$ as

$$\mathbf{E}[\widetilde{R}] \; = \; \mathbf{E}\left[\sum_{m=1}^{T} \widetilde{R}_m\right] \; = \; \mathbf{E}\left[\sum_{m=1}^{T} \mathbf{E}[\widetilde{R}_m \mid T_m]\right] \; \ge \; \frac{\epsilon}{12}\,\mathbf{E}\left[\sum_{m=1}^{T} T_m\right] \; = \; \frac{\epsilon T}{12} \; .$$

Hence, by Lemma 2.15, the expected regret of the player can be lower-bounded as

$$\mathbf{E}[R] \; \ge \; \left(\frac{1}{12} - \frac{1}{25}\right) \cdot \epsilon T \; \ge \; \frac{\epsilon T}{30} \; .$$

Using our choice of ϵ given in (2) concludes the proof. ∎

2.5. Lower bound for stateful policies. In this section we use the lower bound proved in section 2.4 in the hidden bandit setting to prove a similar lower bound in the stateful policies model. Our result applies even in a very restricted case of the problem, where the reference set Π consists of *reactive* policies (see Definition 1.2). This result is stated in Theorem 1.4, repeated here in a more specific form.

THEOREM 1.4 (restated). *For any randomized algorithm in the stateful policies model, there exists a set Π of $k = 3$ reactive policies over $n = 3$ actions, and a sequence* *of oblivious reward functions* $r_1, \dots, r_T$*, such that expected regret of the algorithm with respect to* Π *is* $\Omega(T/\log^{3/2} T)$*.*

Proof. Assume the contrary, namely, that there is an algorithm A that achieves an expected regret of $o(T/\log^{3/2} T)$ with respect to any set of three reactive policies over three actions, and for any oblivious sequence of reward functions. We show that A can be used to achieve the same expected regret in the hidden bandit model with $p = 1/2$ and any oblivious assignment of rewards to the arms. More specifically, we design an algorithm A' for the hidden bandit problem based on the algorithm A that obtains expected regret of $o(T/\log^{3/2} T)$. This would contradict Theorem 1.6, which states that such algorithm cannot exist, proving our claim.

Consider an instance of the hidden bandit problem with $p = 1/2$ and an arbitrary sequence of oblivious reward functions $r'_{1:T} : \{0, 1\} \mapsto [0, 1]$. We now describe a set of reference reactive policies $\Pi = \{\pi_1, \pi_2, \pi_3\}$ and a randomized construction of reward functions $r_{1:T}$ over actions $\{1, 2, 3\}$ that simulates the hidden bandit instance. For convenience, we will construct functions $r_{1:T}$ that assign reward values in the range $[-3, 3]$; this only affects the constants in the resulting bounds.

Reference policies. The reference set Π consists of three reactive policies, π_1, π_2, π_3, that all share the same action function

$$\pi : [-3, 3] \mapsto \{1, 2, 3\} , \qquad \pi(r) = \lfloor |r| \rfloor ,$$

mapping the last observed reward to the next action. The policies differ only by their initial action on round $t = 1$, where policy π_i begins by playing action i.

Reward functions. To construct the functions $r_{1:T}$, we draw a sequence of permutations $\sigma_1, \dots, \sigma_{T+1}$ chosen independently and uniformly at random from the set of all permutations over the elements $\{1, 2, 3\}$ and define for each action $i \in \{1, 2, 3\}$ the following reward sequence:

$$(8) \qquad \forall\, t \in [T] \qquad r_t(i) \;=\; \mathrm{round}\big(r'_t\left(\mathbb{1}_{i \neq \sigma_t(1)}\right),\; \sigma_{t+1}(\sigma_t^{-1}(i))\big) ,$$

where round(r, j) is a randomized rounding function that rounds a reward value $r \in [-1, 1]$ to $\pm j$ in a way that $\mathbf{E}[\mathrm{round}(r, j)] = r$, namely,

$$\mathrm{round}(r, j) \;=\; \begin{cases} +j & \text{with probability} \quad \frac{1}{2}(1 + r/j) , \\ -j & \text{with probability} \quad \frac{1}{2}(1 - r/j) . \end{cases}$$

In particular, there are only six possible reward values: $\pm 1, \pm 2, \pm 3$.

Algorithm. Finally, we define an algorithm A' for the hidden bandit problem, based on A. Let $X_1, \dots, X_T \in \{1, 2, 3\}$ denote the sequence of actions played by A on the reward functions $r_{1:T}$ and the reference set Π. Then, on any round in which A follows the function π, that is, whenever $X_{t+1} = \pi(r_t(X_t))$, the algorithm A' chooses stay; otherwise, it chooses switch. The arm being pulled by A' on round t is given by the random variable $X'_t = \mathbb{1}_{X_t \neq \sigma_t(1)}$, and its reward on that round is $r'(X'_t)$.

The transition function g and the reward functions $r_{1:T}$ together define three disjoint random paths of actions throughout the game, each corresponding to one of the policies π_1, π_2, π_3. Namely, for each $i = 1, 2, 3$, the policy $\pi_{\sigma_1(i)}$ (which plays the action $\sigma_1(i)$ on the first round) plays the sequence of actions $\sigma_1(i), \sigma_2(i), \dots, \sigma_T(i)$ on rounds $1, 2, \dots, T$, which is disjoint from the trajectory of other policies. This follows from the observation that, for any $i \in \{1, 2, 3\}$,

$$(9) \qquad \forall\, t \in [T] \qquad \pi\big(r_t(\sigma_t(i))\big) \;=\; \sigma_{t+1}(i) .$$

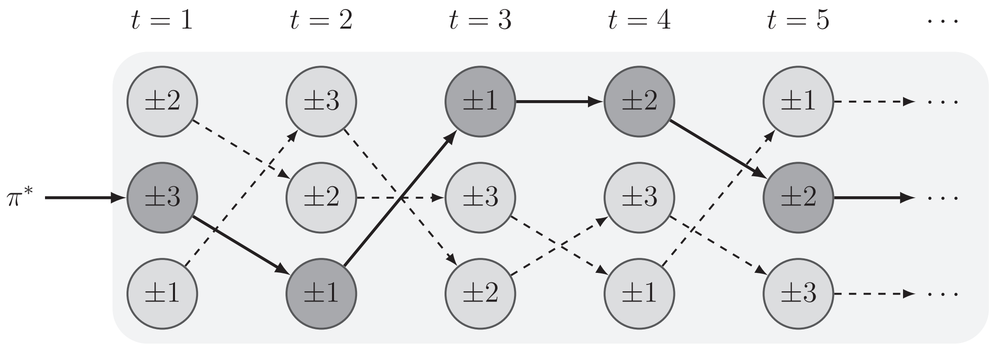


FIG. 7. *An illustration of the reward functions and policies used in the reduction. The marked path represents the path of the policy π^* that corresponds to the reference arm in the hidden bandit problem. The absolute value of each rounded reward on one of the paths indicates the next action on that path.*

In words, if the action $\sigma_t(i)$ was played on round t, then by following the function π the action $\sigma_{t+1}(i)$ is played on round $t+1$. The idea is that the type of rounding used for each reward value on one of the paths signals the function π which is the next action to be played on that path. See Figure 7 for an illustration of this structure.

The policy $\pi^* = \pi_{\sigma_1(1)}$, whose action on the first round is $\sigma_1(1)$, corresponds to the reference arm in the underlying hidden bandit problem. Indeed, by (8), the expected sequence of rewards encountered along the path of π^* is $r'_1(0), r'_2(0), \ldots, r'_T(0)$ (where the expectation is taken with respect to the randomized rounding) and thus coincides with the reward sequence of the reference arm. The expected reward sequences corresponding to the other two paths are identical to the reward sequence of the decoy arm.

We now show that A' is a valid algorithm in the hidden bandit model (with $p = 1/2$). To this end, we verify that the dynamics of the stay and switch actions are compatible with those of the hidden bandit model:

(i) The arm pulled by A' on the first round is the reference arm with probability 1/3 (and it is the decoy arm with probability 2/3). Indeed, $\Pr(X'_1 = 0) = \Pr(X_1 = \sigma_1(1)) = 1/3$ since σ_1 is chosen uniformly at random.

(ii) If A' chooses stay on round t, then it remains on the same arm on round $t+1$, namely, $X'_{t+1} = X'_t$ with probability 1. Indeed, A' chooses stay if $X_{t+1} = \pi(r_t(X_t))$, in which case (9) implies that $X_{t+1} = \sigma_{t+1}(1)$ if and only if $X_t = \sigma_t(1)$, so $X'_{t+1} = X'_t$.

(iii) If A' chooses switch on round t and $X'_t = 0$, then $X'_{t+1} = 1$ with probability 1. This holds since $X'_t = 0$ means that $X_t = \sigma_t(1)$, and if A' chose switch, then necessarily $X_{t+1} \neq \sigma_{t+1}(1)$, which implies that $X'_{t+1} = 1$ with probability 1.

(iv) If A' chooses switch on round t and $X'_t = 1$, then $X'_{t+1} = 0$ with probability 1/2. To see this, note that $X'_t = 1$ implies that $X_t = \sigma_t(j)$ for some $j \neq 1$, so if A' chose switch, then $X_{t+1} \neq \sigma_{t+1}(j)$ and consequently $\Pr(X_{t+1} = \sigma_{t+1}(1)) = 1/2$ as σ_{t+1} is a random permutation. This means that $X'_{t+1} = 0$ with probability 1/2.

Finally, notice that by (8) we have $\mathbf{E}[r'_t(X'_t)] = \mathbf{E}[r_t(X_t)]$ for all t. Together with the fact that the total expected reward of π^* equals the total expect reward of the reference arm, this implies that the expected regret of A' (with respect to the reference arm) is no more than the expected regret of A (with respect to Π), which is assumed to be $o(T/\log^{3/2} T)$. As explained earlier, this contradicts Theorem 1.6 and as a consequence proves our claim. $\square$

2.6. Semi-Markovian players versus consistent adversaries. In this section we consider some natural restrictions on the hidden bandits setting. Some of these restrictions limit the adversary, whereas others limit the player.

We first describe two types of restrictions on the adversary, one being more severe than the other. Recall that the adversary determines the sequence of rewards for each arm. We use $r_t(0)$ ($r_t(1)$, respectively) to denote the reward of the reference arm (decoy arm, respectively) at round t and assume that $0 \le r_t(i) \le 1$.

Definition 2.17 (consistent adversary). *An adversary is* constant *if for every arm i, there is a certain value $v_i \in [0,1]$ such that $r_t(i) = v_i$ for every $1 \le t \le T$. Hence the strategy of a constant adversary is limited to selecting two values $0 \le v_1 < v_0 \le 1$. An adversary is* consistent *if there is a fixed offset $0 \le \Delta \le 1$ such that in every round t, $r_t(0) - r_t(1) = \Delta$. Hence the strategy of a consistent adversary is limited to selecting an offset value $0 \le \Delta \le 1$ and an arbitrary sequence of rewards $r_t(0)$ for the reference arm, in the range $[\Delta, 1]$.*

When the adversary is consistent, the regret of a player is exactly Δ times the number of rounds spent on the decoy arm. Every constant adversary is also a consistent adversary, with $\Delta = v_0 - v_1$.

Let us now present two types of restrictions on the algorithm of the player, one being more severe than the other. Recall that an algorithm of a player determines for every round t whether to switch or stay, depending on the history observable to the player up to and including round t (namely, the sequence or rewards observed, and the time steps of all previous switch requests). It is convenient to assume a round 0 that contains an initial switch action (there is no reward in round 0).

Definition 2.18 (Markovian algorithm). *An algorithm of the player is* Markovian *if there is a deterministic function $p : [0,1] \mapsto [0,1]$, which given a reward r received in round t, maps it to a probability $p(r)$ for switching in round $t+1$. Hence, a Markovian strategy ignores all information available to the player (including the round number), except for the last reward received.*

Definition 2.19 (semi-Markovian algorithm). *An algorithm is* semi-Markovian *if it depends only on the sequence of rewards obtained since the last switch request. Namely, its input is a* memory string *s starting by a switch request, and then continuing with a nonempty sequence of rewards (observed since the last switch up to and including the current round), and its output is a probability $p(s)$ of switching. Then the player tosses a coin with bias $p(s)$. If it comes up heads the player makes a* switch *request and consequently "forgets" the current memory string s and starts building a new memory string by placing a* switch *at its beginning. If the coin comes up tails the player chooses* stay *and keeps the current memory string s. In either case, the reward observed in the next round will be appended to the memory string.*

2.6.1. Upper bound for consistent adversaries. We now discuss a simple algorithm for the hidden bandit problem, suitable for the case where the adversary is playing a consistent strategy. The algorithm, given in Algorithm 4, simply chooses switch with probability inversely proportional to the exponent of the last observed reward.

The intuition behind this algorithm is straightforward: since one arm is constantly better than the other, the player is more likely to switch when he is on the worse arm.

Theorem 2.20. *For $\eta = \frac{1}{2}\log T$, the expected regret of Algorithm 4 is $O\left(\frac{T}{p\log T}\right)$ against any consistent adversary.*

- For $t = 1, \ldots, T$: observe reward r_t and switch with probability $\frac{1}{2}e^{-\eta r_t}$ (otherwise stay)

Algorithm 4: An algorithm for the hidden bandit problem with a consistent adversary, that keeps a constant gap between the rewards sequences of the two arms (parameters: η).

Proof of Theorem 2.20. Let X_t denote the arm assigned to the algorithm on round t. Denote by q_i^t the probability that the algorithm requests a switch on round t given that it is pulling arm i ($i = 0, 1$). Then

$$\begin{aligned} \Pr(X_{t+1} = 1 \mid X_t = 0) &= q_0^t \ , \\ \Pr(X_{t+1} = 0 \mid X_t = 1) &= pq_1^t \ . \end{aligned}$$

Hence, we can view the sequence $X_1, X_2, \ldots, X_T$ as a trajectory of a two-state Markov chain, with its transition kernel on time t given by

$$Q_t = \begin{pmatrix} 1 - q_0^t & q_0^t \\ pq_1^t & 1 - pq_1^t \end{pmatrix} .$$

Our goal is to prove that this chain mixes quickly to a steady-state distribution. Note, however, that the chain is *time inhomogeneous* (there is a different transition matrix on each round), so standard bounds on mixing times do not apply. Nevertheless, our analysis hinges on the fact that there exists a single distribution which is stationary with respect to all transition kernels simultaneously and shows that the chain mixes to that distribution.

First, we identify the steady-state distribution shared by all transition kernels.

LEMMA 2.21. *The distribution*

$$\mu = \left(\frac{p}{p + e^{-\eta\Delta}} \ , \ \frac{e^{-\eta\Delta}}{p + e^{-\eta\Delta}} \right) \tag{10}$$

is stationary with respect to all kernels P_t. That is, it holds that $\mu P_t = \mu$ for all t.

Next, we prove that the inhomogeneous chain mixes to the common stationary distribution μ. In the following, we let μ^t denote the distribution of X_t, namely, the probability distribution over the arms induced by the algorithm on round t.

LEMMA 2.22. *For all t, we have $\|\mu^t - \mu\|_1 \le 2(1 - pe^{-\eta})^{t-1}$.*

The proofs of both lemmas are deferred to the end of the section. Now, note that the expected regret of the player can be written in terms of the distributions μ^t as

$$\mathbf{E}[R_T] = \Delta \cdot \sum_{t=1}^{T} \mu_1^t \ .$$

Suppose that the player could sample directly from the stationary distribution μ (on each round independently). Then, his expected regret would be

$$\Delta \cdot \sum_{t=1}^{T} \mu_1 = T \cdot \frac{\Delta e^{-\eta\Delta}}{p + e^{-\eta\Delta}} \le \frac{T}{\eta p} \cdot \eta\Delta e^{-\eta\Delta} \le \frac{T}{2\eta p} \ ,$$

where in the last inequality we have used the fact that the function $x \mapsto xe^{-x}$ for $x \ge 0$ is maximized at $x = 1$. Using Lemma 2.22, we can bound the difference between this regret and the player's actual regret:

$$\Delta \cdot \sum_{t=1}^{T} (\mu_1^t - \mu_1) \;\le\; \sum_{t=1}^{T} \|\mu^t - \mu\|_1 \;\le\; \sum_{t=1}^{\infty} 2(1 - pe^{-\eta})^{t-1} \;=\; \frac{2e^{\eta}}{p} \;.$$

Overall, we have

$$\mathbf{E}[R_T] \;\le\; \frac{2e^{\eta}}{p} + \frac{T}{2\eta p} \;.$$

Choosing $\eta = \frac{1}{2}\log T$ we obtain

$$\mathbf{E}[R_T] \;\le\; \frac{1}{p}\left(2\sqrt{T} + \frac{T}{\log T}\right) \;=\; O\left(\frac{T}{p\log T}\right) \;,$$

as claimed. $\square$

Finally, we provide the proofs of the lemmas used in our analysis above.

Proof of Lemma 2.21. A simple calculation verifies that the distribution

$$\nu_t \;=\; \left(\frac{pq_1^t}{q_0^t + pq_1^t} \;,\; \frac{q_0^t}{q_0^t + pq_1^t}\right)$$

is stationary with respect to P_t. However, observe that

$$\frac{q_0^t}{q_1^t} \;=\; \frac{e^{-\eta r_t(0)}}{e^{-\eta r_t(1)}} \;=\; e^{-\eta(r_t(0) - r_t(1))} \;=\; e^{-\eta\Delta} \;,$$

which gives

$$\nu_t \;=\; \left(\frac{p}{p + q_0^t/q_1^t} \;,\; \frac{q_0^t/q_1^t}{p + q_0^t/q_1^t}\right) \;=\; \left(\frac{p}{p + e^{-\eta\Delta}} \;,\; \frac{e^{-\eta\Delta}}{p + e^{-\eta\Delta}}\right) \;=\; \mu$$

for all t. $\square$

For the proof of Lemma 2.22 we need the following technical result, which is a variant of Theorem 4.9 in [15].

LEMMA 2.23. *Let Q be a $k \times k$ stochastic matrix such that $Q_{ij} \ge \epsilon$ for some $\epsilon > 0$ and all i, j. Then for any two distribution vectors μ and ν we have*

$$\|\mu Q - \nu Q\|_1 \;\le\; (1 - k\epsilon) \cdot \|\mu - \nu\|_1 \;.$$

Proof. Since $Q_{i,j} \ge \epsilon$ for all i, j, we can write $Q = (1 - k\epsilon)M + \epsilon J$, where M is a stochastic matrix and J denotes the all-ones $k \times k$ matrix. Now, for any two distributions μ, ν we have $\mu J = \nu J = \mathbf{1}$. Consequently,

$$\|\mu Q - \nu Q\|_1 \;=\; (1 - k\epsilon) \cdot \|(\mu - \nu)M\|_1 \;.$$

It remains to bound the norm on the right-hand side. We have

$$\begin{aligned}
\|(\mu - \nu)M\|_1 \;&=\; \sum_{j=1}^{k} \Big| \sum_{i=1}^{k} (\mu_i - \nu_i) M_{ij} \Big| \\
&\le\; \sum_{i=1}^{k}\sum_{j=1}^{k} |\mu_i - \nu_i| M_{ij} \;=\; \sum_{i=1}^{k} |\mu_i - \nu_i| \sum_{j=1}^{k} M_{ij} \;=\; \sum_{i=1}^{k} |\mu_i - \nu_i| \\
&=\; \|\mu - \nu\|_1 \;,
\end{aligned}$$

which completes the proof. $\square$

Proof of Lemma 2.22. Since $q_0^t, q_1^t \le \frac{1}{2}$, the smallest entry in the matrix Q_t is pq_1^t, which is bounded from below by

$$pq_1^t \;=\; \tfrac{1}{2}pe^{-\eta r_t(1)} \;\ge\; \tfrac{1}{2}pe^{-\eta}$$

as the reward $r_t(1)$ is at most one. Hence, Lemma 2.23 above implies that for all t,

$$\|\mu^{t+1} - \mu\|_1 \;=\; \|\mu^t Q_t - \mu Q_t\|_1 \;\le\; (1 - pe^{-\eta}) \cdot \|\mu^t - \mu\|_1 \;,$$

where we have also used the stationarity of π. By repeating this argument, we get

$$\|\mu^{t+1} - \mu\|_1 \;\le\; (1 - pe^{-\eta})^t \cdot \|\mu^1 - \mu\|_1 \;\le\; 2(1 - pe^{-\eta})^t$$

since the L_1-distance between two distributions is at most 2. ◻

2.6.2. Lower bound for semi-Markovian algorithms. We refer to the following constant strategy for the adversary as the MT strategy. MT stands for *monotonicity testing*, as this strategy (and the first part of the analysis of Case 2 in the proof of Theorem 2.24 below) is a variation on a procedure of Raskhodnikova [20] for testing whether a one-variable function is monotone. For simplicity and with only negligible affect on the end results, assume that $1 + \log T$ is a power of 2.

The constant strategy MT. The adversary chooses at random two integer values $1 \le k_1 < k_0 \le \log T$, subject to the following constraint. Let r be the largest power of 2 that divides either k_1 or k_0 (whichever gives a larger value for r). Then the constraint is that $k_0 - k_1 \le 2^r$. For $0 \le r \le \log\log T$, say that a pair $k_1 < k_0$ is in *class* r if $2^{r-1} < k_0 - k_1 \le 2^r$. Observe that for each value of r there are at most $\log T$ pairs (k_1, k_0) in class r. Let $c = \sum_{r=0}^{\log\log T} \frac{1}{(r+1)^2}$ and observe that $c < 2$ (regardless of T). The probability distribution from which the pair (k_1, k_0) is chosen is as follows: first an integer value $0 \le r \le \log\log T$ is chosen with probability $\frac{1}{c(r+1)^2}$. Then a pair (k_1, k_0) from class r is chosen uniformly at random. Finally, given the chosen k_1 and k_0, the adversary sets $v_0 = \frac{k_0}{\log T}$ and $v_1 = \frac{k_1}{\log T}$.

THEOREM 2.24. *In the hidden bandit setting, regardless of the value of the parameter p, if the player is restricted to using semi-Markovian strategies, then his expected regret against the* MT *strategy is* $\Omega(T/\log T)$.

Proof. To obtain a lower bound on the regret it suffices to consider deterministic strategies for the player, because the strategy of the adversary is already fixed to be MT. Given that each arm has constant reward under the MT strategy, then a deterministic semi-Markovian strategy of the player is simply a function $g : [0,1] \mapsto \{1, 2, \ldots, T\}$ that maps the observed reward r to how many rounds should elapse since the previous switch until the next switch. Consider an arbitrary such strategy g of the player, and consider the function $f(k) = g(k/\log T)$, defined on integer k in the range $[1, \log T]$. Let $\mathsf{LIS}(f)$ be the length of the longest monotone increasing subsequence of the sequence $f(1), f(2), \ldots, f(\log T)$. We consider two cases.

Case 1: $\mathsf{LIS}(f) \ge \frac{4}{5}\log T$. Let $i_1, \ldots, i_\ell$ be the indices of an increasing subsequence of length $\ell \ge \frac{4}{5}\log T$. For at most $\frac{1}{2}\log T$ values of j we have that $f(i_{j+1}) \ge 4f(i_j)$, as otherwise $f(i_\ell) > 4^{\log T/2} f(i_1) = Tf(i_1) \ge T$, which is outside the range of the function p. Observe also that the increasing subsequence, being of length at least $\frac{4}{5}\log T$, must contain at least $\frac{3}{5}\log T$ consecutive pairs, where a consecutive pair is a value j such that $i_{j+1} = 1 + i_j$. Hence at least $\frac{1}{10}\log T$ consecutive pairs differ by a factor less than 4. Namely, there are at least $\frac{1}{10}\log T$ values of i for which $f(i+1) \le 4f(i)$. Refer to such a pair $(i, i+1)$ as a *dangerous pair*. The dangerous pair is in class 0 (with classes as defined in the MT strategy). The probability of the MT strategy to choose class 0 is $\frac{1}{c} \geq \frac{1}{2}$, and if chosen, the dangerous pair is chosen with probability $\frac{1}{\log T}$. On the dangerous pair, the expected number of rounds the player spend on the decoy arm (the one with reward $\frac{i}{\log T}$ rather than $\frac{i+1}{\log T}$) is at least $\frac{T}{5}$, because $f(i+1) \leq 4f(i)$. On the decoy arm the regret is $\frac{1}{\log T}$. Hence the expected regret is at least

$$\frac{1}{10}\log T \cdot \frac{1}{2} \cdot \frac{1}{\log T} \cdot \frac{T}{5} \cdot \frac{1}{\log T} \;=\; \frac{T}{100 \log T} \;.$$

Case 2: $\mathsf{LIS}(f) < \frac{4}{5}\log T$. Consider a graph with vertices labeled from 1 to $\log T$, where an edge connects vertex i to vertex $j > i$ if and only if they form a *decreasing pair* with respect to f, namely, $f(j) < f(i)$. Consider a maximal matching in this graph. Its size is at least $\frac{1}{10}\log T$, because otherwise all vertices not involved in a maximal matching would form an increasing subsequence longer than $\frac{4}{5}\log T$. Consider an arbitrary matching edge (i, j), and let h in the range $i \leq h \leq j$ be such that the power of two that divides it is highest. If $h \in \{i, j\}$, then the pair (i, j) is a possible choice of the MT algorithm. We call this an MT-pair. If $i < h < j$, then both (i, h) and (h, j) are MT-pairs, and at least one of them is decreasing. Hence each matching edge contributes at least one decreasing MT-pair. A simple case analysis (which is omitted) shows that two different matching edges cannot possibly contribute the same decreasing MT-pair. Hence there are at least $\frac{1}{10}\log T$ decreasing MT-pairs. Observe that these pairs need not be disjoint: the same h vertex can participate in several (or even all) pairs.

Let c be the constant in the definition of the MT strategy. Then there must be a choice of integer r in the range $0 \leq r \leq \log\log T$ that is *dangerous* in the sense that there are $\frac{\log T}{10c(r+1)^2}$ decreasing MT-pairs of class r. This dangerous r is chosen with probability $\frac{1}{c(r+1)^2}$. Given that a dangerous r is chosen, the probability of choosing a decreasing MT-pair of class r is at least $\frac{1}{10c(r+1)^2}$, because there are at most $\log T$ pairs in a class. If a decreasing MT-pair is selected, the player spends in expectation at least $T/2$ steps on the decoy arm. In these steps, the regret of the player is at least $\frac{2^{r-1}}{\log T}$. Hence the expected regret is at least (using also $c < 2$):

$$\frac{1}{c(r+1)^2} \cdot \frac{1}{10c(r+1)^2} \cdot \frac{T}{2} \cdot \frac{2^{r-1}}{\log T} \;\geq\; \frac{2^{r+1}T}{320(r+1)^4 \log T} \;.$$

The expression $\frac{2^{r+1}}{(r+1)^4}$ is minimized when $r = 5$, giving roughly $\frac{1}{20}$. Hence the expected regret in case 2 is at most $\frac{T}{7000 \log T}$.

We have not attempted to optimize the leading constant in the Ω notation, and no doubt it can be substantially improved beyond the bounds shown in our proof. ∎

The combination of Theorems 2.20 and 2.24 implies the following result.

Theorem 1.7 (restated). *Consider a hidden bandit problem with $p = \frac{1}{2}$, where the adversary is restricted to be consistent and the player is restricted to use semi-Markovian algorithms. Then, the player can guarantee expected regret $O(T/\log T)$, and this is the best possible guarantee in this setting. Moreover, to achieve this guarantee Markovian strategies suffice, and to block stronger guarantees constant adversaries suffice.*

2.7. Reactive adversaries. In this section we discuss an extension of our basic setting, where the adversary is allowed to be *reactive*. Reactive adversaries are adversaries that set the reward given at any round as a function of the actions of the player on that and the ℓ previous rounds (for some fixed ℓ). In other words, a reactive adversary determines a sequence of reward functions $r_1, \ldots, r_T : [n]^{\ell+1} \mapsto [0,1]$ before the game begins, where each function in the sequence is used to map $\ell+1$ consecutive actions of the player to a reward value. With the notation of section 1.1, the expected regret of the player in this case is given by

$$\text{Regret}_T = \max_{\pi \in \Pi} \sum_{t=1}^{T} r_t(x_t^\pi, \ldots, x_{t-\ell}^\pi) - \mathbf{E}\left[\sum_{t=1}^{T} r_t(X_t, \ldots, X_{t-\ell})\right] .$$

For simplicity, we focus on the case where $\ell = 1$.

When the reference policies are stateless, previous work showed that the player can obtain $\widetilde{O}(T^{2/3})$ expected regret against a reactive adversary [1], and this rate is best possible in general [6]. These results do not hold when the reference policies are stateful; indeed, our lower bound (Theorem 1.4) clearly extends to reactive adversaries as the oblivious adversary we have considered above is a special case of a reactive adversary.

Our algorithms can be adapted to the reactive adversary setting. We sketch our approach, while omitting the technical details. For simplicity we assume that $\ell = 1$, though the approach easily extends to arbitrary values of ℓ, at a cost of higher upper bounds on the regret as ℓ grows. Essentially, the only issue we have to address is providing a new implementation of a switch action in Algorithm 3 that switches at some round t to a random policy and initializes it in a random state. The difficulty is that even if the switch is successful and results in the best policy at its correct state for round t, the reward observed as feedback on round t is also a function of the action in round $t-1$, where a different policy was followed (and a possibly different feedback was observed). Hence applying the state transition function with this reward might cause the policy to reach an incorrect state at round $t+1$. We deal with this problem by spending two rounds, t and $t+1$, on implementing the switch operation. In round t the player plays a random action. In round $t+1$ the player picks a random policy to switch to and guesses its internal state (as done by Algorithm 3) and plays the action recommended by the guessed policy in the guessed state. With probability $1/(knS)$ all the following conditions hold simultaneously: the player guessed the best reference policy at its correct state for round $t+1$, and the actions performed in rounds t and $t+1$ are exactly as would have been chosen had the player followed the best policy all along. Thereafter, the feedback received in round $t+1$ puts the player on the right track of the best policy. This approach retains the sublinear regret rate provided by Algorithm 3 (in terms of T), but the dependence on the constants involves also n, and not only k and s.

Appendix A. Martingales and locally repetitive strings. Lemma 2.4 is central to our work. Hence, it is instructive to see another proof for it. Let us recall for this purpose Doob's upcrossing inequality for martingales. Let $X_1, \ldots, X_n$ be a martingale, namely, a sequence of random variables such that $X_i = E[X_{i+1} \mid X_1, \ldots, X_i]$ for all i, and suppose that the range of values of the martingale is bounded in the sense that $0 \leq X_n \leq 1$ (hence the same necessarily holds for all X_i). Fixing $0 \leq a < b \leq 1$, an (a,b)-upcrossing in a sequence is a pair of indices $i < j$ such that $X_i \leq a$ and $X_j \geq b$. The number of (a,b)-upcrossings in the sequence is the largest number t such that there are indices $i_1 < j_i < i_2 < j_2 \ldots < i_t < j_t$, and for every $1 \leq \ell \leq t$, there is an (a,b) upcrossing in (i_ℓ, j_ℓ). The following lemma is known as

Doob's upcrossing inequality for martingales, for which we sketch a simple proof for completeness.

Lemma A.1 (Doob's upcrossing inequality). *For the setting as above, the expected number of* (a, b) *upcrossing is at most* $a/(b - a)$.

*Proof (*sketch*).* Think of X_i as the value of a stock at time i. Due to the martingale property, there is no trading policy for the stock that gains money in expectation. Consider the strategy of buying the stock as soon as it drops below a, selling it as soon as it moves above b, and so on. At the last round the stock is sold regardless of its value. If the number of crossings is U, this strategy makes a profit of $(b - a)U$. Selling on the last round loses at most a (the maximum possible value at which the stock was bought). Hence $(b - a)\mathbf{E}[U] - a \leq 0$, proving the lemma. ∎

Fix $\epsilon > 0$ that divides 1, and for integer $0 \leq m < 1/\epsilon$, we refer to an $(m\epsilon, (m+1)\epsilon)$ upcrossing as an ϵ-upcrossing. Summing over all m, Lemma A.1 implies that the expected number of ϵ-upcrossings is

$$\frac{1}{\epsilon}\sum_{m=1}^{1/\epsilon}(m-1)\epsilon \;<\; \frac{1}{2\epsilon^2}\ .$$

A similar bound applies by symmetry to the analogous notion of ϵ-downcrossing. Hence altogether the expected number of ϵ crossings is at most $1/\epsilon^2$.

We can now sketch an alternative proof for Lemma 2.4 (with somewhat weaker bounds).

Proof of Lemma 2.4 *(*sketch*).* Starting from s, consider the process of partitioning s into d substrings and choosing one of them at random, and doing so recursively until a single character u is reached. The averages $x_s, \ldots, x_u$ encountered on such a random path form a martingale sequence, with final value in $[0, 1]$. In expectation, at most $4/\epsilon^2$ of the steps were an $\epsilon/2$-crossing. Observe that for every string v that is encountered along the way, if v is not (d, ϵ)-repetitive, then there is probability at least $1/d$ of moving to a substring u that inflicts an $\epsilon/2$ crossing (the inequality $|x_v - x_u| > \epsilon$ implies that within this range there is a crossing of width $\epsilon/2$ aligned at a multiple of $\epsilon/2$). Hence at most $4d/\epsilon^2$ steps went through strings that are not (d, ϵ)-repetitive. As there are k steps, this implies that $\delta \leq 4d/k\epsilon^2$. ∎

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), 2012, pp. 1503–1510.

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

[3] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E Schapire, *The nonstochastic multiarmed bandit problem*, SIAM J. Comput., 32 (2002), pp. 48–77.

[4] N. Cesa-Bianchi, Y. Freund, D. Haussler, D. P. Helmbold, R. E. Schapire, and M. K. Warmuth, *How to use expert advice*, J. ACM, 44 (1997), pp. 427–485.

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

[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), 2014, pp. 459–467.

[7] O. Dekel and E. Hazan, *Better rates for any adversarial deterministic MDP*, in Proceedings of the 30th International Conference on Machine Learning, 2013, pp. 675–683.

[8] A. Drucker, *High-confidence predictions under adversarial uncertainty*, in Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ACM, 2012, pp. 1–10.

[9] 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 (STOC), ACM, 2010, pp. 715–724.

[10] E. Even-Dar, S. M. Kakade, and Y. Mansour, *Online Markov decision processes*, Math. Oper. Res., 34 (2009), pp. 726–736.

[11] D. Pucci De Farias and N. Megiddo, *Combining expert advice in reactive environments*, J. ACM, 53 (2006), pp. 762–799.

[12] M. Feder, N. Merhav, and M. Gutman, *Universal prediction of individual sequences*, IEEE Trans. Inform. Theory, 38 (1992), pp. 1258–1270.

[13] U. Feige, *Why are images smooth?*, in Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ACM, 2015, pp. 229–236.

[14] Y. Freund and R. E. Schapire, *A decision-theoretic generalization of on-line learning and an application to boosting*, J. Comput. System Sci., 55 (1997), pp. 119–139.

[15] D. A. Levin, Y. Peres, and E. L. Wilmer, *Markov Chains and Mixing Times*, AMS, Providence, RI, 2009.

[16] N. Merhav and M. Feder, *Universal prediction*, IEEE Trans. Inform. Theory, 44 (1998), pp. 2124–2147.

[17] N. Merhav, E. Ordentlich, G. Seroussi, and M. J. Weinberger, *On sequential strategies for loss functions with memory*, IEEE Trans. Inform. Theory, 48 (2002), pp. 1947–1958.

[18] G. Neu, A. György, C. Szepesvári, and A. Antos, *Online Markov decision processes under bandit feedback*, in Advances in Neural Information Processing Systems 23, MIT Press, Cambridge, MA, 2010, pp. 1804–1812.

[19] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic Game Theory*, Vol. 1, Cambridge University Press, Cambridge, UK, 2007.

[20] S. Raskhodnikova, *Monotonicity Testing*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, 1999.

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