Appearing in proceedings, 36th Annual Symposium on Foundations of Computer Science, November, 1995.

Electronic Colloquium on Computational Complexity, Report No. 68 (2000)

Gambling in a rigged casino: The adversarial multi-armed bandit problem

Peter Auer
Computer & Information Sciences
University of California
Santa Cruz, CA 95064
pauer@cse.ucsc.edu

Nicolò Cesa-Bianchi
DSI
Università di Milano
20135 Milano (Italy)
cesabian@dsi.unimi.it

Yoav Freund
Robert E. Schapire
AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974
{yoav, schapire}@research.att.com

Abstract

In the multi-armed bandit problem, a gambler must decide which arm of K non-identical slot machines to play in a sequence of trials so as to maximize his reward. This classical problem has received much attention because of the simple model it provides of the trade-off between exploration (trying out each arm to find the best one) and exploitation (playing the arm believed to give the best payoff). Past solutions for the bandit problem have almost always relied on assumptions about the statistics of the slot machines.

In this work, we make no statistical assumptions whatsoever about the nature of the process generating the payoffs of the slot machines. We give a solution to the bandit problem in which an adversary, rather than a well-behaved stochastic process, has complete control over the payoffs. In a sequence of T plays, we prove that the expected per-round payoff of our algorithm approaches that of the best arm at the rate $O(T^{-1/3})$, and we give an improved rate of convergence when the best arm has fairly low payoff.

We also consider a setting in which the player has a team of "experts" advising him on which arm to play; here, we give a strategy that will guarantee expected payoff close to that of the best expert. Finally, we apply our result to the problem of learning to play an unknown repeated matrix game against an all-powerful adversary.

1 Introduction

In the well studied multi-armed bandit problem, originally proposed by Robbins [9], a gambler must choose which of K slot machines to play. At each time step, he pulls the arm of one of the machines and receives a reward or payoff (possibly zero or negative). The gambler's purpose is to maximize his total reward over a sequence of trials. Since each arm is assumed to have a different distribution of rewards, the goal is to find the arm with the best expected return as early as possible, and then to keep gambling using that arm.

The problem is a classical example of the trade-off between exploration and exploitation. On the one hand, if the gambler plays exclusively on the machine that he thinks is best ("exploitation"), he may fail to discover that one of the other arms actually has a higher average return. On the other hand, if he spends too much time trying out all the machines and gathering statistics ("exploration"), he may fail to play the best arm often enough to get a high total return.

As a more practically motivated example, consider the task of repeatedly choosing a route for transmitting packets between two points in a communication network. Suppose there are K possible routes and the transmission cost is reported back to the sender. Then the problem can be seen as that of selecting a route for each packet so that the total cost of transmitting a large set of packets would not be much larger than the cost incurred by sending them all on the single best route.

In the past, the bandit problem has almost always been studied with the aid of statistical assumptions on the process generating the rewards for each arm. In the gambling example, for instance, it might be natural to assume that the distribution of rewards for each arm is Gaussian and time-invariant. However, it is likely that the costs associated with each route in the routing example cannot be modeled by a stationary distribution, so a more sophisticated set of statistical assumptions would be required. In general, it may be difficult or impossible to determine the right statistical assumptions for a given domain, and some domains may be inherently adversarial in nature so that no such assumptions are appropriate.

In this paper, we present a variant of the bandit problem in which *no* statistical assumptions are made about the generation of rewards. In our model, the reward associated with each arm is determined at each time step by an adversary with unbounded computational power rather than by some benign stochastic process. We only assume that the rewards are chosen from a bounded range. The performance of any player is measured in terms of *regret*, i.e., the difference between the total reward scored by the player and the total reward scored by the best arm.

At first it may seem impossible that the player should stand a chance against such a powerful opponent. Indeed, a *deterministic* player will fare very badly against an adversary who assigns low payoff to the chosen arm and high payoff

ISSN 1433-8092

to all the other arms. However, in this paper we present a very efficient, randomized player algorithm that performs well against any adversary. We prove that the regret suffered by our algorithm is at most $O(T^{2/3}(K \log K)^{1/3})$, where K is the number of arms and T is the number of time steps. Note that the average per-time-step regret approaches zero at the rate $O(T^{-1/3})$.

We also present more refined bounds in which the dependence on T is replaced by the total reward of the best arm (or an assumed upper bound thereof).

Our worst-case bounds may appear weaker than the bounds proved using statistical assumptions, such as those shown by Lai and Robbins [6] of the form $O(\log T)$. However, when comparing our results to those in the statistics literature, it is important to point out two differences between our framework and theirs:

1. They define the regret as the difference between the expected total reward of the player and the maximum of the expected total rewards of any arm. Our definition, in contrast, measures regret with respect to the specific sequence of payoffs actually generated by the adversary.

2. They assume that the distribution of rewards that is associated with each arm is fixed as the number of iterations T increases to infinity. In contrast, our bounds hold for any finite T, and, by the generality of our model, these bounds are applicable when the payoffs are randomly (or adversarially) chosen in a manner that does depend on T.

While it might seem that the apparent weakness of our bounds is a result of the adversarial nature of our framework, in fact, each of the differences described above suffices to show that an upper bound of $O(\log T)$ is impossible in our model. This holds even when the rewards are generated randomly and independently of the player's actions as in the standard statistical framework.

Consider the first difference. In a statistical setting, the difference in the definition of the regret corresponds to the difference between the maximum expected total reward of any arm and the expected maximum total reward of any arm in a sequence of T trials. These two measures can be far apart since, due to random variation, the expected maximum is typically much larger than the maximal expectation. As shown by Cesa-Bianchi et al. [2], this idea can be used to construct a lower bound for the regret of any algorithm of the form $\Omega(\sqrt{T \log K})$.[1]

We prove a stronger lower bound in Section 6 that is based on the second difference. We describe a distribution over the rewards of the different arms, which depends on T, for which the regret of any player is $\Omega(\sqrt{TK})$.

A non-stochastic bandit problem was also considered by Gittins [4] and Ishikida and Varaiya [5]. However, their version of the bandit problem is very different from ours: they assume that the player can compute ahead of time exactly what payoffs will be received from each arm, and their problem is thus one of optimization, rather than exploration and exploitation.

Our algorithm is based in part on an algorithm recently presented by Freund and Schapire [3], which in turn is a variant of Littlestone and Warmuth's [7] weighted majority algorithm, and Vovk's [10] aggregating strategies. In the setting analyzed by Freund and Schapire (which we call here the *full information game*), the player on each trial scores the reward of the chosen arm, but gains access to the rewards associated with *all* of the arms (not just the one that was chosen).

In some situations picking the same action at all trials might not be the best strategy. For example, in the packet routing problem it might be that no single route is good for the whole duration of the message, but switching between routes from time to time can achieve a better performance. We give a variant of our algorithm which combines the choices of N strategies (or "experts"), each of which recommends one of the K actions at each iteration. We show that the regret with respect to the best strategy is $O(T^{2/3}(K \ln N)^{1/3})$. Note that the dependence on the number of strategies is only logarithmic, and therefore the bound is quite reasonable even when the player is combining a very large number of strategies.

The adversarial bandit problem is closely related to the problem of learning to play an unknown repeated matrix game. In this setting, a player without prior knowledge of the game matrix is playing the game repeatedly against an adversary with complete knowledge of the game and unbounded computational power. It is well known that matrix games have an associated *value* which is the best possible expected payoff when playing the game against an adversary. If the matrix is known, then a randomized strategy that achieves the value of the game can be computed (say, using a linear-programming algorithm) and employed by the player. The case where the matrix is entirely unknown was previously considered by Baños [1] and Megiddo [8], who proposed two (extremely inefficient) strategies whose per-round payoff converges to the game value. For the same problem, we show that by using our algorithm the player achieves an expected per-round payoff in T rounds which efficiently approaches the value of the game at the rate $O(T^{-1/3})$. This convergence is much faster than that achieved by Baños and Megiddo.

Our paper is organized as follows. In Section 2, we give the formal definition of the problem. In Section 3, we describe Freund and Schapire's algorithm for the full information game and state its performance. In Section 4, we describe our basic algorithm for the partial information game. In Section 5, we show how to adaptively tune the parameters of this algorithm when no prior knowledge is available. In Section 6, we give a lower bound on the regret suffered by any algorithm for the partial information game. In Section 7, we show how to modify the algorithm to use expert advice. Finally, in Section 8, we describe the application of our algorithm to

[1] In fact, this lower bound holds for the stronger full information game described in Section 3.

repeated matrix games.

2 Notation and terminology

We formalize the bandit problem as a game between a player choosing actions and an adversary choosing the rewards associated with each action. The game is parameterized by the number K of possible actions, where each action is denoted by an integer i, $1 \leq i \leq K$. We will assume that all the rewards belong to the unit interval $[0, 1]$. The generalization to rewards in $[a, b]$ for arbitrary $a < b$ is straightforward.

The game is played in a sequence of trials $t = 1, 2, \ldots, T$. We distinguish two variants: the partial information game, which captures the adversarial multi-armed bandit problem; and the full information game, which is essentially equivalent to the framework studied by Freund and Schapire [3]. On each trial t of the *full information game*:

1. The adversary selects a vector $\mathbf{x}(t) \in [0, 1]^K$ of current rewards. The ith component $x_i(t)$ is interpreted as the reward associated with action i at trial t.

2. Without knowledge of the adversary's choice, the player chooses an action by picking a number $i_t \in \{1, 2, \ldots, K\}$ and scores the corresponding reward $x_{i_t}(t)$.

3. The player observes the entire vector $\mathbf{x}(t)$ of current rewards.

The *partial information game* corresponds to the above description of the full information game but with step 3 replaced by:

3'. The player observes only the reward $x_{i_t}(t)$ for the chosen action i_t.

Let $G_A \doteq \sum_{t=1}^{T} x_{i_t}(t)$ be the total reward of player A choosing actions $i_1, i_2, \ldots, i_T$.

We formally define an adversary as a deterministic rule mapping the past history of play $i_1, \ldots, i_{t-1}$ to the current reward vector $\mathbf{x}(t)$. (Since all our results are worst-case with respect to the adversary, there is no additional power to be gained by allowing the adversary to be randomized). As a special case, we say that an adversary is *oblivious* if it is independent of the player's actions, i.e., if the reward at trial t is a function of t only. Clearly, all of our results, which are proved for a non-oblivious adversary, hold for an oblivious adversary as well.

As our player algorithms will be randomized, fixing an adversary and a player algorithm defines a probability distribution over the set $\{1, \ldots, K\}^T$ of sequences of T actions. All the probabilities and expectations considered in this paper will be with respect to this distribution. For an oblivious adversary, the rewards are fixed quantities with respect to this distribution, but for a non-oblivious adversary, each reward $x_i(t)$ is a random variable defined on the set $\{1, \ldots, K\}^{t-1}$ of player actions up to trial $t - 1$. We will not use explicit notation to represent this dependence, but will refer to it in the text when appropriate.

The measure of the performance of our algorithm is the *regret*, which is the expected value of the difference between the total reward of the algorithm and the total reward of the best action. Formally, we define the expected total reward of algorithm A by

$$\mathbf{E}[G_A] \doteq \mathbf{E}_{i_1,\ldots,i_T}\left[\sum_{t=1}^{T} x_{i_t}(t)\right],$$

the expected total reward of the best action by

$$G_{\text{best}} \doteq \max_{1 \leq j \leq K} \mathbf{E}_{i_1,\ldots,i_T}\left[\sum_{t=1}^{T} x_j(t)\right],$$

and the regret of algorithm A by $R_A \doteq \mathbf{E}[G_A] - G_{\text{best}}$. This definition is easiest to interpret for an oblivious adversary since, in this case, G_{best} truly measures what could have been gained had the best action been played for the entire sequence. However, for a nonoblivious adversary, the definition of regret is a bit strange: Although it still compares the total reward of the algorithm to the sum of rewards that were associated with taking some action j on all iterations, had action j been taken, the rewards chosen by the adversary would have been different than those actually generated, since the variable $x_j(t)$ depends on the past history of plays $i_1, \ldots, i_{t-1}$. Although the definition of R_A looks difficult to interpret in this case, in Section 8 we prove that our bounds on the regret for a nonoblivious adversary can also be used to derive an interesting result in the context of repeated matrix games.

3 The full information game

In this section, we describe an algorithm, called $\mathbf{Hedge}$, for the full information game which will also be used as a building block in the design of our algorithm for the partial information game. The version of $\mathbf{Hedge}$ presented here is a slight variant[2] of the algorithm introduced by Freund and Schapire [3] as a generalization of Littlestone and Warmuth's Weighted Majority [7] algorithm.

$\mathbf{Hedge}$ is described in Figure 1. The main idea is simply to choose action i at time t with probability proportional to $(1 + \alpha)^{s_i(t)}$, where $\alpha > 0$ is a parameter and $s_i(t)$ is the total reward scored so far by action i. Thus, actions yielding high rewards quickly gain a high probability of being chosen.

The following is a straightforward variant of Freund and Schapire's Theorem 2. For completeness, a proof is provided in Appendix A.

[2]These modifications enable $\mathbf{Hedge}$ to handle gains (rewards in $[0, 1]$) rather than losses (rewards in $[-1, 0]$).

Algorithm Hedge
Parameter: A real number $\alpha > 0$.
Initialization: Set $s_i(1) := 0$ for $i = 1, \ldots, K$.

Repeat for $t = 1, 2, \ldots$ until game ends

1. Choose action i_t according to the distribution $\mathbf{p}(t)$, where
$$p_i(t) = \frac{(1+\alpha)^{s_i(t)}}{\sum_{j=1}^{K}(1+\alpha)^{s_j(t)}}.$$
2. Receive the reward vector $\mathbf{x}(t)$ and score gain $x_{i_t}(t)$.
3. Set $s_i(t+1) := s_i(t) + x_i(t)$ for $i = 1, \ldots, K$.

Figure 1: Algorithm $\mathbf{Hedge}$ for the full information game.

Lemma 3.1 *For $\alpha > 0$, and for any sequence of reward vectors $\mathbf{x}(1), \ldots, \mathbf{x}(T)$, the probability vectors $\mathbf{p}(t)$ computed by* **Hedge** *satisfy*

$$\sum_{t=1}^{T} \mathbf{p}(t) \cdot \mathbf{x}(t) \geq \frac{\left(\sum_{t=1}^{T} x_j(t)\right) \ln(1+\alpha) - \ln K}{\alpha}$$

for all actions $j = 1, \ldots, K$.

Taking expectations with respect to the random choice of plays and using standard approximations for the ln function gives a lower bound on the expected gain:

Theorem 3.2 *For $\alpha > 0$, the expected gain of algorithm* **Hedge** *in the full information game is at least*

$$\begin{aligned} \mathbf{E}[G_{\mathbf{Hedge}}] &\geq \frac{G_{\text{best}} \ln(1+\alpha) - \ln K}{\alpha} \\ &\geq G_{\text{best}} - \frac{\alpha}{2} G_{\text{best}} - \frac{\ln K}{\alpha}. \end{aligned}$$

Thus, it can easily be shown that, for an appropriate choice of α, $\mathbf{Hedge}$ suffers regret at most $\sqrt{2T \ln K}$ in the full information game.

4 The partial information game

In this section, we move to the analysis of the partial information game. We present an algorithm $\mathbf{Exp3}$ that runs the algorithm $\mathbf{Hedge}$ of Section 3 as a subroutine. ($\mathbf{Exp3}$ stands for "Exponential-weight algorithm for Exploration and Exploitation.")

The algorithm is described in Figure 2. On each trial t, $\mathbf{Exp3}$ receives the distribution vector $\mathbf{p}(t)$ from $\mathbf{Hedge}$, and selects an action i_t according to the distribution $\hat{\mathbf{p}}(t)$ which is a mixture of $\mathbf{p}(t)$ and the uniform distribution. Intuitively,

Algorithm Exp3
Parameters: Reals $\alpha > 0$ and $\gamma \in [0, 1]$.
Initialization: Initialize $\mathbf{Hedge}$.

Repeat for $t = 1, 2, \ldots$ until game ends

1. Get the distribution $\mathbf{p}(t)$ from $\mathbf{Hedge}$.
2. Select action i_t to be j with probability $\hat{p}_j(t) = (1-\gamma)p_j(t) + \gamma/K$.
3. Receive reward $x_{i_t}(t) \in [0, 1]$.
4. Feed the simulated reward vector $\hat{\mathbf{x}}(t)$ back to $\mathbf{Hedge}$, where
$$\hat{x}_j(t) = \begin{cases} \frac{\gamma}{K} \cdot \frac{x_{i_t}(t)}{\hat{p}_{i_t}(t)} & \text{if } j = i_t \\ 0 & \text{otherwise.} \end{cases}$$

Figure 2: Algorithm $\mathbf{Exp3}$ for the partial information game.

mixing in the uniform distribution is done in order to make sure that the algorithm tries out all K actions and gets good estimates of the rewards for each. Otherwise, the algorithm might miss a good action because the initial rewards it observes for this action are low and large rewards that occur later are not observed because the action is not selected.

After $\mathbf{Exp3}$ receives the reward $x_{i_t}(t)$ associated with the chosen action, it generates a simulated reward vector $\hat{\mathbf{x}}(t)$ for $\mathbf{Hedge}$. As $\mathbf{Hedge}$ requires full information, all components of this vector must be filled in, even for the actions that were not selected. For actions $j \neq i_t$ not chosen, we set $\hat{x}_j(t)$ to be zero. For the chosen action i_t, we set the simulated reward $\hat{x}_{i_t}(t)$ proportional to $x_{i_t}(t)/\hat{p}_{i_t}(t)$. This compensates the reward of actions that are unlikely to be chosen and guarantees that the expected simulated gain associated with any fixed action j is proportional to the actual gain of the action, i.e., that $\mathbf{E}_{i_t}[\hat{x}_j(t) \mid i_1, \ldots, i_{t-1}] = \frac{\gamma}{K} x_j(t)$ for any fixed choice of $i_1, \ldots, i_{t-1}$. The constant scaling factor γ/K is used to guarantee that the rewards fed back to $\mathbf{Hedge}$ are in the range $[0, 1]$ as required. From this perspective, it is necessary to mix in the uniform distribution with the distribution generated by $\mathbf{Hedge}$ in order to ensure that the simulated reward before scaling, $x_{i_t}(t)/\hat{p}_{i_t}(t)$, is not too large.

We now give the main theorem of this paper, which bounds the regret of algorithm $\mathbf{Exp3}$.

Theorem 4.1 *For $\alpha > 0$, $\gamma \in [0, 1]$, the expected gain of algorithm* **Exp3** *is at least*

$$\mathbf{E}[G_{\mathbf{Exp3}}] \geq \frac{1-\gamma}{\alpha}\left(G_{\text{best}} \ln(1+\alpha) - \frac{K \ln K}{\gamma}\right)$$

$$\geq \quad G_{\text{best}} - \left(\gamma + \frac{\alpha}{2}\right) G_{\text{best}} - \frac{K \ln K}{\alpha\gamma}.$$

By making an appropriate choice of the parameters γ and α we get the following bound.

Corollary 4.2 *If $g \geq G_{\text{best}}$ and algorithm* **Exp3** *is run with input parameters: $\alpha = \sqrt[3]{(4K \ln K)/g}$ and $\gamma = \min\left\{1, \sqrt[3]{(K \ln K)/(2g)}\right\}$, then the regret of algorithm* **Exp3** *is at most*

$$R_{\mathbf{Exp3}} \leq \frac{3}{\sqrt[3]{2}} g^{2/3} (K \ln K)^{1/3}.$$

Proof of Theorem 4.1. We begin by showing a lower bound on the total reward $\sum_t x_{i_t}(t)$ for *any* sequence $i_1, \ldots, i_t$:

$$\begin{aligned}
\sum_{t=1}^{T} x_{i_t}(t) &= \frac{K}{\gamma} \sum_{t=1}^{T} \hat{p}_{i_t}(t) \hat{x}_{i_t}(t) \\
&= \frac{K}{\gamma} \sum_{t=1}^{T} \left((1-\gamma) p_{i_t}(t) \hat{x}_{i_t}(t) + \frac{\gamma}{K} \hat{x}_{i_t}(t) \right) \\
&\geq \frac{(1-\gamma)K}{\gamma} \sum_{t=1}^{T} p_{i_t}(t) \hat{x}_{i_t}(t) \\
&= \frac{(1-\gamma)K}{\gamma} \sum_{t=1}^{T} \mathbf{p}(t) \cdot \hat{\mathbf{x}}(t) \qquad (1) \\
&\geq \frac{(1-\gamma)K}{\gamma\alpha} \left(\ln(1+\alpha) \sum_{t=1}^{T} \hat{x}_j(t) - \ln K \right). \qquad (2)
\end{aligned}$$

Equation (2) holds for any action j by Lemma 3.1 since the simulated reward vectors $\hat{\mathbf{x}}(t)$ are fed back to $\mathbf{Hedge}$ at each time step. For any j, t we have

$$\begin{aligned}
\mathbf{E}[\hat{x}_j(t)] &= \mathbf{E}_{i_1,\ldots,i_{t-1}} \left[\mathbf{E}_{i_t} [\hat{x}_j(t) \mid i_1, \ldots, i_{t-1}] \right] \\
&= \mathbf{E}_{i_1,\ldots,i_{t-1}} \left[\hat{p}_j(t) \cdot \frac{\gamma}{K} \cdot \frac{x_j(t)}{\hat{p}_j(t)} + (1 - \hat{p}_j(t)) \cdot 0 \right] \\
&= \frac{\gamma}{K} \mathbf{E}[x_j(t)]. \qquad (3)
\end{aligned}$$

Thus, taking expectations in (2), we get that

$$\begin{aligned}
&\mathbf{E}\left[\sum_{t=1}^{T} x_{i_t}(t) \right] \\
&\geq \frac{(1-\gamma)K}{\gamma\alpha} \left[\frac{\gamma}{K} \left(\sum_{t=1}^{T} \mathbf{E}[x_j(t)] \right) \ln(1+\alpha) - \ln K \right].
\end{aligned}$$

As the inequality holds for all j, we get the bound of the theorem. The bound on regret follows from the fact that $\ln(1+\alpha) \geq \alpha - \alpha^2/2$ for $\alpha > 0$. □

To apply Corollary 4.2, it is necessary that an upper bound g on G_{best} be available for tuning α and γ. For example, if the number of trials T is known in advance then, since no action can have payoff greater than 1 on any trial, we can use $g = T$ as an upper bound.

If the rewards $x_i(t)$ are in the range $[a, b]$, $a < b$, then **Exp3** can be used after the rewards have been translated and rescaled to the range $[0, 1]$. Applying Corollary 4.2 with $g = T$, this gives a bound on regret of the form $O((b-a)T^{2/3}(K \ln K)^{1/3})$. For instance, this is applicable to a standard loss model where the "rewards" fall in the range $[-1, 0]$.

Finally, in our definition of the game, we have assumed that the game always ends after T trials. However, we can get the same bounds on the regret even if we allow the adversary to end the game after an arbitrary trial of the adversary's choosing. The reason is that this case can be reduced to the previous one by having the adversary generate reward zero on all rounds after the original game ends.

Algorithm Exp3.1
Initialization: Set $n := 1$ and $\tilde{s}_i := 0$ for $i = 1, \ldots, K$.

1. Set $g(n) := 2^n$. Restart **Exp3** and let the parameters α and γ of **Exp3** be as in Corollary 4.2 (with g set to $g(n)$).
2. Let **Exp3** choose an action i_t. After the reward $x_{i_t}(t)$ is received, update $\tilde{s}_{i_t} := \tilde{s}_{i_t} + x_{i_t}(t)/\hat{p}_{i_t}(t)$.
3. If $\max_i \tilde{s}_i > g(n) - K/\gamma$ Then set $n := n + 1$ and **goto** 1. Else **goto** 2.

Figure 3: Algorithm **Exp3.1** for the partial information game when a bound on G_{best} is not known.

5 Guessing the maximal reward

In the last section, we showed that algorithm **Exp3** yields a regret of $O(g^{2/3}(K \ln K)^{1/3})$ whenever an upper bound g on the total expected reward G_{best} of the best action is known in advance. In this section, we describe an algorithm **Exp3.1** which does not require prior knowledge of a bound on G_{best}, and whose regret is at most $O(G_{\text{best}}^{2/3}(K \ln K)^{1/3})$ for $G_{\text{best}} = \Omega(K^3)$.

Our algorithm **Exp3.1**, described in Figure 3, proceeds in *rounds*, where each round consists of a sequence of trials. We use $n = 1, 2, \ldots$ to index the rounds. On round n, the algorithm "guesses" a bound $g(n)$ for the total reward of the best action. It then uses this guess to tune the parameters α and γ of **Exp3**. On iteration t, after **Exp3** chooses action i_t and receives reward $x_{i_t}(t)$, the estimate $\tilde{s}_{i_t}$ of the total reward of action i_t is incremented by $x_{i_t}(t)/\hat{p}_{i_t}(t)$. Dividing the

reward by the probability of taking the action guarantees that the expected value of each estimate is correct, i.e., $\mathbf{E}[\tilde{s}_j(t)] = \sum_{t'=1}^{t} x_j(t')$ for all $1 \le j \le K$ and all $1 \le t \le T$, where $\tilde{s}_j(t)$ denotes the value of $\tilde{s}_j$ at the end of trial t. Using these estimates, the algorithm detects (approximately) when the actual gain of some action has advanced beyond $g(n)$. When this happens, the algorithm increments n and restarts **Exp3** with a larger bound on the maximal gain.

The performance of the algorithm is characterized by the following theorem.

Theorem 5.1 *For $K \ge 2$, the regret suffered by algorithm* **Exp3.1** *is at most*

$$R_{\textbf{Exp3.1}} \le 43(G_{\text{best}}^{2/3}\sqrt[3]{K \ln K} + K^2\sqrt[3]{\ln K}).$$

(We did not attempt to optimize the constants in this theorem.)

The proof of the theorem is divided into two lemmas. The first bounds the regret caused by each round, and the second bounds the number of rounds. We define the following random variables: T_n denotes the trial on which the nth round begins and R denotes the total number of rounds (i.e., the value of n on the last trial T). For notational convenience, we also define T_{R+1} to be $T+1$, and we define $\hat{y}_j(t) = \frac{K}{\gamma}\hat{x}_j(t)$ for $1 \le j \le K$, $1 \le t \le T$. Note that $\tilde{s}_j(t) = \sum_{t'=1}^{t} \hat{y}_j(t')$.

Lemma 5.2 *For any action j and for every round $1 \le n \le R$, the gain of* **Exp3.1** *during the nth iteration is lower bounded by*

$$\sum_{t=T_n}^{T_{n+1}-1} x_{i_t}(t) \ge \sum_{t=T_n}^{T_{n+1}-1} \hat{y}_j(t) - g(n)^{2/3}\frac{3}{\sqrt[3]{2}}\sqrt[3]{K \ln K}.$$

Proof. We use Equation (2) from the proof of Theorem 4.1. We replace $\frac{K}{\gamma}\hat{x}_j(t)$ by $\hat{y}_j(t)$ and separate the term involving $\sum_{t=T_n}^{T_{n+1}-1} \hat{y}_j(t)$ into two terms as follows:

$$\begin{aligned}
&\sum_{t=T_n}^{T_{n+1}-1} x_{i_t}(t) \\
&\quad \ge \frac{(1-\gamma)K}{\alpha\gamma}\left(\ln(1+\alpha)\sum_{t=T_n}^{T_{n+1}-1}\hat{x}_j(t) - \ln K\right) \\
&\quad \ge \sum_{t=T_n}^{T_{n+1}-1}\hat{y}_j(t) - \left(\gamma + \frac{\alpha}{2}\right)\sum_{t=T_n}^{T_{n+1}-1}\hat{y}_j(t) - \frac{K\ln K}{\alpha\gamma}.
\end{aligned}$$

We bound the second occurrence of the sum by adding non-negative terms: $\sum_{t=T_n}^{T_{n+1}-1}\hat{y}_j(t) \le \sum_{t=1}^{T_{n+1}-1}\hat{y}_j(t) = \tilde{s}_j(T_{n+1}-1)$. From the definition of the termination condition, we know that $\tilde{s}_j(T_{n+1}-1) \le g(n)$. Substituting this bound and our choices for α and γ into the last equation above we get the statement of the lemma. □

The next lemma shows that, with high probability, there are not too many rounds.

Lemma 5.3 *Let $K \ge 2$, $G_{\text{best}} \ge K$, and $r = \lceil \log_2 G_{\text{best}} \rceil$. Then for any $i \ge 3$, $\mathbf{P}\{R > r+i\} \le 10.2 \cdot 2^{-5i/3}K^{5/3}/G_{\text{best}}^{2/3}$.*

Proof. (Sketch) The idea of the proof is simple. If algorithm **Exp3.1** terminated round number $r+i$ before iteration T, then the value of at least one of the K estimators $\tilde{s}_1, \ldots, \tilde{s}_K$ at iteration T has to be larger than $g(r+i) - K/\gamma(r+i)$, where $\gamma(n)$ is the value of the parameter γ used on round n. However, the expected value of the estimator is $G_{\text{best}} \le g(r)$, so reaching round $r+i$ implies a large estimation error. Simple arguments from probability theory show that the probability of such a large error in any of the estimators is very small. (Details given in Appendix B.) □

Proof of Theorem 5.1. Since the theorem holds trivially when $G_{\text{best}} \le K$, we assume without generality that $G_{\text{best}} \ge K$.

We fix some action j, partition the expected total gain into runs, and bound the gain from each round using Lemma 5.2:

$$\begin{aligned}
\mathbf{E}\left[\sum_{t=1}^{T} x_{i_t}(t)\right] &= \mathbf{E}\left[\sum_{n=1}^{R}\sum_{t=T_n}^{T_{n+1}-1} x_{i_t}(t)\right] \\
&\ge \mathbf{E}\left[\sum_{n=1}^{R}\sum_{t=T_n}^{T_{n+1}-1}\hat{y}_j(t)\right] \\
&\quad -3\sqrt[3]{\frac{K\ln K}{2}}\,\mathbf{E}\left[\sum_{n=1}^{R} g(n)^{2/3}\right] \\
&\ge \mathbf{E}\left[\sum_{t=1}^{T} x_j(t)\right] \\
&\quad -6.5\sqrt[3]{K\ln K}\,\mathbf{E}\left[2^{2R/3}\right].
\end{aligned}$$

If we now choose j to be the action with the largest gain, then the first term is equal to G_{best} and the second term bounds the regret. To bound the second term we separate the expectation into the sum of a typical term and the atypical terms and then use Lemma 5.3.

$$\begin{aligned}
\mathbf{E}\left[2^{2R/3}\right] &\le 2^{2(r+3)/3}\mathbf{P}\{R \le r+3\} \\
&\quad + \sum_{i=3}^{\infty} 2^{2(r+i+1)/3}\mathbf{P}\{R > r+i\} \\
&\le (16G_{\text{best}})^{2/3} \\
&\quad + (4G_{\text{best}})^{2/3} \cdot 10.2 \cdot \frac{K^{5/3}}{G_{\text{best}}^{2/3}}\sum_{i=3}^{\infty} 2^{-i} \\
&\le 6.5G_{\text{best}}^{2/3} + 6.5K^{5/3}.
\end{aligned}$$

This gives the statement of the theorem. □

6 A lower bound

In this section, we prove an information-theoretic lower bound on the regret of any player, i.e., a lower bound that holds

even if the player has unbounded computational power. More precisely, we show that there exists an adversarial strategy for choosing the rewards such that the expected regret of any player algorithm is $\Omega(\sqrt{TK})$. Observe that this does not match the upper bound for our algorithms $\mathbf{Exp3}$ and $\mathbf{Exp3.1}$ (see Corollary 4.2 and Theorem 5.1); it is an open problem to close this gap.

The adversarial strategy we use in our proof is oblivious to the algorithm; it simply assigns the rewards at random according to some distribution, similar to a standard statistical model for the bandit problem. The choice of distribution depends on the number of actions K and the number of iterations T. This dependence of the distribution on T is the reason that our lower bound does not contradict the upper bounds of the form $O(\log T)$ which appear in the statistics literature [6]. There, the distribution over the rewards is fixed as $T \to \infty$.

For the full information game, matching upper and lower bounds of the form $\theta\left(\sqrt{T \log K}\right)$ were already known [2]. Our lower bound shows that for the partial information game the dependence on the number of actions increases considerably. Specifically, our lower bound implies that no upper bound is possible of the form $O(T^{\alpha}(\log K)^{\beta})$ where $0 \leq \alpha < 1$, $\beta > 0$.

Theorem 6.1 *There exist some positive constant $c > 0$ and natural number K_0 such that for any number of actions $K \geq K_0$ and any number of iterations $T \geq K$ there exists a distribution over the rewards assigned to different actions such that the expected regret of any algorithm is at least $c\sqrt{TK}$.*

The lower bound on the expected regret implies, of course, that for any algorithm there is a particular choice of rewards that will cause the regret to be larger than this expected value.

Proof. (Sketch) We construct the random distribution of the rewards as follows. One of the K actions is chosen uniformly at random to be the "good" action. The T rewards associated with the good action are chosen independently at random to be 1 with probability $1/2 + a\sqrt{K/T}$ and 0 otherwise for some small constant $a > 0$. The rewards associated with the other actions are chosen independently at random to be 0 or 1 with equal odds. Then the average optimal reward per trial is $1/2 + a\sqrt{K/T}$. On the other hand, we show that the difference between the distributions of rewards of good and bad arms is so slight that, in T trials, the algorithm cannot detect which is the good arm with sufficient reliability. More precisely, there is a constant probability that the good action is sampled only $2T/K$ times and the total gain is at most $T/2$. From this a lower bound, on the expected regret can be derived. (Details omitted for lack of space.) □

7 Combining the advice of many experts

Up to this point, we have considered a bandit problem in which the player's goal is to achieve a payoff close to that of the best single action. In a more general setting, the player may have a set of strategies for choosing the best action. These strategies might select different actions at different iterations. The strategies can be computations performed by the player or they can be external advice given to the learner by "experts." We will use the more general term "expert" (borrowed from Cesa-Bianchi et al. [2]) because we place no restrictions on the generation of the advice. The player's goal in this case is to combine the advice of the experts in such a way that its total reward is close to that of the best expert (rather than the best single action).

For example, consider the packet routing problem. In this case there might be several routing strategies, each based on different assumptions regarding network load distribution and using different data to estimate current load. Each of these strategies might suggest different routes at different times, and each might be better in different situations. In this case, we would like to have an algorithm for combining these strategies which, for each set of packets, performs almost as well as the strategy that was best for that set.

Formally, at each trial t, we assume that the player, prior to choosing an action, is provided with a set of N probability vectors $\boldsymbol{\xi}^j(t) \in [0,1]^K$, $j = 1, \ldots, N$, $\sum_{i=1}^{K} \xi_i^j(t) = 1$. We interpret $\boldsymbol{\xi}^j(t)$ as the advice of expert j on trial t, where the ith component $\xi_i^j(t)$ represents the recommended probability of playing action i (as a special case, the distribution can be concentrated on a single action, which represents a deterministic recommendation). If the adversary chooses payoff vector $\mathbf{x}(t)$, then the expected reward for expert j (with respect to the chosen probability vector $\boldsymbol{\xi}^j(t)$) is simply $\boldsymbol{\xi}^j(t) \cdot \mathbf{x}(t)$. In analogy of G_{best}, we define

$$\tilde{G}_{\text{best}} \doteq \max_{1 \leq j \leq N} \mathbf{E}_{i_1, \ldots, i_T} \left[\sum_{t=1}^{T} \boldsymbol{\xi}^j(t) \cdot \mathbf{x}(t) \right],$$

so that the regret $\tilde{R}_A \doteq \mathbf{E}[G_A] - \tilde{G}_{\text{best}}$ measures the expected difference between the player's total reward and the total reward of the best expert.

Our results hold for any finite set of experts. Formally, we regard each $\boldsymbol{\xi}^j(t)$ as a random variable, which is an arbitrary function of the random sequence of plays $i_1, \ldots, i_{t-1}$ (just like the adversary's payoff vector $\mathbf{x}(t)$). This definition allows for experts whose advice depends on the entire past history as observed by the player, as well as other side information which may be available.

We could at this point view each expert as a "meta-action" in a higher-level bandit problem with payoff vector defined at trial t as $(\boldsymbol{\xi}^1(1) \cdot \mathbf{x}(t), \ldots, \boldsymbol{\xi}^t(N) \cdot \mathbf{x}(t))$. We could then immediately apply Corollary 4.2 to obtain a bound of $O(g^{2/3}(N \log N)^{1/3})$ on the player's regret relative to the best expert (where g is an upper bound on $\tilde{G}_{\text{best}}$). However, this bound is quite weak if the player is combining many experts (i.e., if N is very large).

We show below that the algorithm $\mathbf{Exp3}$ from Sec-

Algorithm Exp4
Parameters: Reals $\alpha > 0$ and $\gamma \in [0,1]$
Initialization: Initialize $\mathbf{Hedge}$ (with K replaced by N)

Repeat for $t = 1, 2, \ldots$ until game ends

1. Get the distribution $\mathbf{q}(t) \in [0,1]^N$ from $\mathbf{Hedge}$.
2. Get advice vectors $\boldsymbol{\xi}^j(t) \in [0,1]^K$, and let $\mathbf{p}(t) := \sum_{j=1}^N q_j(t)\boldsymbol{\xi}^j(t)$.
3. Select action i_t to be j with probability $\hat{p}_j(t) = (1-\gamma)p_j(t) + \gamma/K$.
4. Receive reward $x_{i_t}(t) \in [0,1]$.
5. Compute the simulated reward vector $\hat{\mathbf{x}}(t)$ as
$$\hat{x}_j(t) = \begin{cases} \frac{\gamma}{K} \cdot \frac{x_{i_t}(t)}{\hat{p}_{i_t}(t)} & \text{if } j = i_t \\ 0 & \text{otherwise.} \end{cases}$$
6. Feed the vector $\mathbf{y}(t) \in [0,1]^N$ to $\mathbf{Hedge}$ where $y_j(t) \doteq \boldsymbol{\xi}^j(t) \cdot \hat{\mathbf{x}}(t)$.

Figure 4: Algorithm $\mathbf{Exp4}$ for using expert advice in the partial information game.

tion 4 can be modified yielding a regret term of the form $O(g^{2/3}(K \log N)^{1/3})$. This bound is very reasonable when the number of actions is small, but the number of experts is quite large (even exponential).

Our algorithm $\mathbf{Exp4}$ is shown in Figure 4, and is only a slightly modified version of $\mathbf{Exp3}$. ($\mathbf{Exp4}$ stands for "Exponential-weight algorithm for Exploration and Exploitation using Expert advice.")

As before, we use $\mathbf{Hedge}$ as a subroutine, but we now apply $\mathbf{Hedge}$ to a problem of dimension N rather than K. At trial t, we receive a probability vector $\mathbf{q}(t)$ from $\mathbf{Hedge}$ which represents a distribution over strategies. We compute the vector $\mathbf{p}(t)$ as a weighted average (with respect to $\mathbf{q}(t)$) of the strategy vectors $\boldsymbol{\xi}^j(t)$. The vector $\hat{\mathbf{p}}(t)$ is then computed as before using $\mathbf{p}(t)$, and an action i_t is chosen randomly. We define the vector $\hat{\mathbf{x}}(t) \in [0,1]^K$ as before, and we finally feed the vector $\mathbf{y}(t) \in [0,1]^N$ to $\mathbf{Hedge}$ where $y_j(t) \doteq \boldsymbol{\xi}^j(t) \cdot \hat{\mathbf{x}}(t)$.

Theorem 7.1 *For $\alpha > 0$, $\gamma \in [0,1]$, and for any family of experts, the expected gain of algorithm* $\mathbf{Exp4}$ *is at least*

$$\begin{aligned} \mathbf{E}[G_{\mathbf{Exp4}}] &\geq \frac{1-\gamma}{\alpha}\left(\tilde{G}_{\text{best}} \ln(1+\alpha) - \frac{K \ln N}{\gamma}\right) \\ &\geq \tilde{G}_{\text{best}} - \left(\gamma + \frac{\alpha}{2}\right)\tilde{G}_{\text{best}} - \frac{K \ln N}{\alpha\gamma}. \end{aligned}$$

Proof. From the definitions above, we have that

$$\mathbf{p}(t) \cdot \hat{\mathbf{x}}(t) = \sum_{j=1}^N q_j(t)\boldsymbol{\xi}^j(t) \cdot \hat{\mathbf{x}}(t) = \mathbf{q}(t) \cdot \mathbf{y}(t).$$

Thus, for all j, using (1) from the proof of Theorem 4.1, and then applying Lemma 3.1, we have

$$\begin{aligned} \sum_{t=1}^T x_{i_t}(t) &\geq \frac{(1-\gamma)K}{\gamma} \sum_{t=1}^T \mathbf{p}(t) \cdot \hat{\mathbf{x}}(t) \\ &= \frac{(1-\gamma)K}{\gamma} \sum_{t=1}^T \mathbf{q}(t) \cdot \mathbf{y}(t) \\ &\geq \frac{(1-\gamma)K}{\alpha\gamma}\left(\ln(1+\alpha)\sum_{t=1}^T y_j(t) - \ln N\right). \end{aligned}$$

Taking expectations and using (3), we see that

$$\mathbf{E}[y_j(t)] = \mathbf{E}[\boldsymbol{\xi}^j(t) \cdot \hat{\mathbf{x}}(t)] = \frac{\gamma}{K}\mathbf{E}[\boldsymbol{\xi}^j(t) \cdot \mathbf{x}(t)]$$

and the theorem follows as in the proof of Theorem 4.1. □

Analogous versions of Corollary 4.2 and Theorem 5.1 can be proved in which $K \ln K$ is replaced in the regret by $K \ln N$.

8 Nearly optimal play of an unknown repeated game

The bandit problem considered up to this point is closely related to the problem of playing an unknown repeated game against an adversary of unbounded computational power. In this latter setting, a game is defined by an $n \times m$ matrix A. On each trial t, the learner (or row player) chooses a row i of the matrix. At the same time, the adversary (column player) chooses a column j. The learner then receives the payoff A_{ij}. In repeated play, the learner's goal is to maximize its expected total payoff over a sequence of plays.

Suppose in some trial the learner chooses its next move i randomly according to a probability distribution on rows represented by a (column) vector $\mathbf{p} \in [0,1]^n$, and the adversary similarly chooses according to a probability vector $\mathbf{q} \in [0,1]^m$. Then the expected payoff is $\mathbf{p}^T A \mathbf{q}$. Von Neumann's famous minimax theorem states that

$$\max_{\mathbf{p}} \min_{\mathbf{q}} \mathbf{p}^T A \mathbf{q} = \min_{\mathbf{q}} \max_{\mathbf{p}} \mathbf{p}^T A \mathbf{q},$$

where the max and min are over all distribution vectors $\mathbf{p}$ and $\mathbf{q}$. The quantity v defined by the above equation is called the *value* of the game given by matrix A. In words, this says that there exists a mixed (randomized) strategy $\mathbf{p}$ for the row player that guarantees expected payoff at least v, regardless of the column player's action. Moreover, this payoff is optimal in the sense that the column player can choose a mixed strategy

whose expected payoff is at most v, regardless of the row player's action.

Thus, if the learner knows the matrix A, it can compute a strategy (for instance, using linear programming) that is certain to bring an expected optimal payoff v on each trial.

Suppose now that the game A is entirely unknown to the learner. To be precise, assume the learner knows only the number of rows of the matrix, and a bound on the magnitude of the entries of A. The main result of this section is a proof based on the results in Section 4 showing that the learner can play in such a manner that its payoff per trial will rapidly converge to the optimal maximin payoff v. This result holds even when the adversary knows the game A, and also knows the (randomized) strategy being used by the learner.

This problem of playing a repeated game with incomplete information was previously considered by Baños [1] and Megiddo [8]. However, these previously proposed strategies are extremely inefficient. Not only is our strategy simpler and much more efficient, but we also are able to prove much faster rates of convergence.

In fact, the application of our earlier algorithms to this problem is entirely straightforward. The learner's actions are now identified with the rows of the matrix and are chosen randomly on each trial according to algorithm **Exp3**, where we tune α and γ as in Corollary 4.2 with $g = T$, where T is the total number of rounds of play.[3] The payoff vector $\mathbf{x}(t)$ is simply the column j_t of A chosen by the adversary on trial t.

Theorem 8.1 *Let A be an unknown game matrix in $[a,b]^{n\times m}$ with value v. Suppose the learner, knowing only a, b and n, uses the algorithm sketched above against any adversary for T trials. Then the learner's expected payoff per trial is at least*

$$v - 3(b-a)\sqrt[3]{\frac{n\ln n}{2T}}.$$

Proof. We assume that $[a,b] = [0,1]$; the extension to the general case is straightforward. By Corollary 4.2, we have

$$\begin{aligned}\mathbf{E}\left[\sum_{t=1}^{T} A_{i_t j_t}\right] &= \mathbf{E}\left[\sum_{t=1}^{T} x_{i_t}(t)\right] \\ &\geq \max_i \mathbf{E}\left[\sum_{t=1}^{T} x_i(t)\right] - \frac{3}{\sqrt[3]{2}} T^{2/3}(n\ln n)^{1/3}.\end{aligned}$$

Let $\mathbf{p}$ be such that $v = \max_{\mathbf{p}} \min_{\mathbf{q}} \mathbf{p}^T A \mathbf{q} = \min_{\mathbf{q}} \mathbf{p}^T A \mathbf{q}$. Then

$$\begin{aligned}\max_i \mathbf{E}\left[\sum_{t=1}^{T} x_i(t)\right] &\geq \sum_{i=1}^{n} p_i \sum_{t=1}^{T} \mathbf{E}\left[x_i(t)\right] \\ &= \sum_{t=1}^{T} \mathbf{E}\left[\mathbf{p}\cdot\mathbf{x}(t)\right] = \sum_{t=1}^{T} \mathbf{E}\left[\mathbf{p}^T A \mathbf{q}_t\right] \geq vT\end{aligned}$$

where $\mathbf{q}_t$ is a distribution vector whose j_tth component is 1. □

[3] If T is not known in advance, the methods developed in Section 5 can be applied.

Note that the theorem is independent of the number of columns of A, and, with appropriate assumptions, the theorem can be easily generalized to adversaries with an infinite number of strategies. If the matrix A is very large and all entries are small then even if A is known to the player, our algorithm may be an efficient alternative to linear programming.

The generality of the theorem also allows us to handle games in which the outcome for given plays i, j is a random variable (rather than a constant $A_{i,j}$). Finally, as pointed out by Megiddo [8], such a result is valid for non-cooperative, multi-person games; the average per-trial payoff of any player using this strategy will converge rapidly to the maximin payoff of the one-shot game.

Acknowledgments

We express special thanks to Kurt Hornik for his advice and his patience when listening to our ideas and proofs for an earlier draft of this paper. We thank Yuval Peres and Amir Dembo for their help regarding the analysis of martingales. Nicolò Cesa-Bianchi gratefully acknowledges support of Esprit NeuroCOLT project No. 8556. Peter Auer acknowledges support of the Fonds zur Förderung der wissenschaftlichen Forschung, Austria, by Schrödinger scholarship J01028-MAT.

References

[1] Alfredo Baños. On pseudo-games. *The Annals of Mathematical Statistics*, 39(6):1932–1945, 1968.

[2] Nicolò Cesa-Bianchi, Yoav Freund, David P. Helmbold, David Haussler, Robert E. Schapire, and Manfred K. Warmuth. How to use expert advice. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 382–391, 1993.

[3] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory: Second European Conference, EuroCOLT '95*, pages 23–37. Springer-Verlag, 1995.

[4] J. C. Gittins. *Multi-armed Bandit Allocation Indices*. John Wiley and Sons, 1989.

[5] T. Ishikida and P. Varaiya. Multi-armed bandit problem revisited. *Journal of Optimization Theory and Applications*, 83(1):113–154, October 1994.

[6] T. L. Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6:4–22, 1985.

[7] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, 1994.

[8] N. Megiddo. On repeated games with incomplete information played by non-Bayesian players. *International Journal of Game Theory*, 9(3):157–167, 1980.

[9] H. Robbins. Some aspects of the sequential design of experiments. *Bulletin American Mathematical Society*, 55:527–535, 1952.

[10] Volodimir G. Vovk. Aggregating strategies. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 371–383, 1990.

A Proof of Lemma 3.1

Let $W_t = \sum_{i=1}^K (1+\alpha)^{s_i(t)}$. For $1 \le t \le T$,

$$\begin{aligned}\frac{W_{t+1}}{W_t} &= \sum_{i=1}^K \frac{(1+\alpha)^{s_i(t)}(1+\alpha)^{x_i(t)}}{W_t} \\ &\le \sum_{i=1}^K \frac{(1+\alpha)^{s_i(t)}(1+\alpha x_i(t))}{W_t} \\ &= 1 + \alpha \frac{\sum_{i=1}^K (1+\alpha)^{s_i(t)} x_i(t)}{W_t}\end{aligned}$$

where we used the fact that $x_i(t) \in [0,1]$ for the inequality. Thus

$$\begin{aligned}\ln \frac{W_{T+1}}{W_1} &= \sum_{t=1}^T \ln \frac{W_{t+1}}{W_t} \\ &\le \sum_{t=1}^T \ln \left(1 + \alpha \frac{\sum_{i=1}^K (1+\alpha)^{s_i(t)} x_i(t)}{W_t}\right) \\ &\le \sum_{t=1}^T \alpha \frac{\sum_{i=1}^K (1+\alpha)^{s_i(t)} x_i(t)}{W_t} \\ &= \alpha \sum_{t=1}^T \sum_{i=1}^K p_i(t) x_i(t).\end{aligned}$$

Observing that $W_1 = K$ and $W_{T+1} \ge (1+\alpha)^{\sum_{t=1}^T x_j(t)}$ for any j, we obtain the statement of the lemma. □

B Proof of Lemma 5.3

To get the upper bound we use

$$\begin{aligned}&\mathbf{P}\{R > r+i\} \\ &= \mathbf{P}\left\{\exists j : \tilde{s}_j(T) > g(r+i) - \frac{K}{\gamma(r+i)}\right\} \\ &\le \sum_{j=1}^K \mathbf{P}\left\{\tilde{s}_j(T) > g(r+i) - \frac{K}{\gamma(r+i)}\right\}\end{aligned}$$

and

$$\begin{aligned}&\mathbf{P}\left\{\tilde{s}_j(T) > g(r+i) - \frac{K}{\gamma(r+i)}\right\} \\ &\le \frac{\mathrm{Var}(\tilde{s}_j(T))}{(g(r+i) - \mathbf{E}[\tilde{s}_j(T)] - \frac{K}{\gamma(r+i)})^2}\end{aligned}$$

by Chebychev's inequality. Since

$$\mathbf{E}[\tilde{s}_j(T)] = \mathbf{E}\left[\sum_{t=1}^T \hat{y}_j(t)\right] = \mathbf{E}\left[\sum_{t=1}^T x_j(t)\right]$$

and the $(\hat{y}_j(t) - x_j(t))$ form a sequence of martingale differences, we have

$$\mathrm{Var}(\tilde{s}_j(T)) = \mathbf{E}\left[\sum_{t=1}^T (\hat{y}_j(t) - x_j(t))^2\right]. \tag{4}$$

Observe that the probability of selecting action j at trial t, given that this trial is within the first $r+i$ rounds, is at least $\mu = \gamma(r+i)/K$. Also observe that increasing the probability of selecting action j to μ on trials *after* $r+i$ rounds are completed does not change the probability of starting $r+i+1$ rounds in T trials. Therefore we can, for the sake of our upper bound, assume that the probability of selecting action j at any trial up to T is at least μ. Evaluating Equation (4), we get

$$\begin{aligned}\mathrm{Var}(\tilde{s}_j(T)) &\le \mathbf{E}\left[\sum_{t=1}^T \frac{(1-\mu)x_j(t)^2}{\mu}\right] \\ &\le \mathbf{E}\left[\sum_{t=1}^T \frac{x_j(t)}{\mu}\right] \le \frac{G_{\text{best}}}{\mu}.\end{aligned}$$

Since $\mathbf{E}[\tilde{s}_j(t)] \le G_{\text{best}}$, $G_{\text{best}} \ge K$, $i \ge 3$ and by our choice of r, it can be verified that

$$g(r+i) - \mathbf{E}[\tilde{s}_j(T)] - \frac{K}{\gamma(r+i)} \ge 0.42 \cdot G_{\text{best}} \cdot 2^i.$$

Thus,

$$\begin{aligned}&\mathbf{P}\left\{\tilde{s}_j(T) \ge g(r+i) - \frac{K}{\gamma(r+i)}\right\} \\ &\le \frac{G_{\text{best}}}{(0.42)^2 \cdot G_{\text{best}}^2 \cdot 2^{2i}\mu} \le 10.2 \left(\frac{K}{G_{\text{best}}}\right)^{2/3} 2^{-5i/3}.\end{aligned}$$

Summing over all actions j gives the lemma. □

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'