Regretful Bandits: A Survey of Algorithms and Regret Bounds for Adversarial Multi-Armed Bandits

Corbin Rosset, Matthew Ige, Edmund Duhaime

Dept. of Computer Science, Johns Hopkins University

Abstract

The Multi-armed Bandit problem is a sequential learning problem where a player chooses an action from a predefined set of actions at every step in time with the objective of maximizing reward, which is formalized as minimization of regret. The MAB problem is important in many practical applications such as packet routing and ad-targeting, and is closely related to reinforcement learning in the artificial intelligence community. In this work we map out the various assumptions, regret objectives, and algorithms for the Multi-armed Bandit (MAB) problem, focusing on the more difficult setting of adversarial (non-stochastic) rewards with only partial information feedback. We make an effort to clarify the numerous definitions of regret that appear in the literature by formulating few of them as special cases of more general notions. Specifically, we explain commonly cited pure-strategy regret (the regret of the best action played repeatedly), worst-case regret (regret against any possible sequence of actions), and external/strategy regret (regret with respect to a class of experts who recommend actions).

Keywords: Multi-armed Bandit, stochasticity

1. Introduction

Sequential learning problems are a class of problems where at each time step t a player selects an element i_t from a finite set of actions, $[K]$. The time horizon, T, of gameplay may or may not be finite. In the multi-armed bandit

Email addresses: crosset2@jhu.edu (Corbin Rosset), mige@jhu.edu (Matthew Ige), eduhaim1@jhu.edu (Edmund Duhaime)

(MAB) problem, the player receives feedback $x(t)$ from some bounded interval on the reals in the form of reward or loss for at least the action she chose (*partial information*), and at most all the actions that were not played (*full information*). In the *stochastic* setting, the rewards are generated randomly and independently from K fixed distributions, however in the *adversarial* MAB setting, there are no assumptions on how the rewards can be generated, and there are ways to quantify how difficult it is to perform against an adversary (*hardness*). However, there may be assumptions about whether the adversary knows what action the player performed, and it may possibly be fully omniscient about the player's beliefs and strategies. Specifically, the *non-oblivious* adversary can generate reward x_t from a function $r(i_1, \ldots, i_{t-1})$ of at least the history of played actions, and possibly more information. An *oblivious* adversary, on the other hand, cannot assign rewards that depend on the current action or on any history of actions; this is equivalent to the adversary fixing the sequence of rewards for each action ahead of time [1, 2].

The objective of the bandit player is to maximize gain $\mathcal{G}$ or minimize loss $\mathcal{L}$, depending on the rewards scheme. The notion of *regret*, denoted $\mathcal{R}$ accumulated by a sequence of actions $\{i_1, i_2, \ldots, i_T\}$ is characterized by how much less cumulative loss the player could have incurred had she played a different sequence of actions $J = \{j_1, j_2, \ldots, j_T\}$. The many definitions of regret restrict what subset of the K^T sequences can be compared, but in all cases the objective is to minize cumulative regret irrespective of rewards assigned to each action. It is commonly desired to achieve "vanishing" or "near zeroregret which is sublinear in T, $o(T)$.

By default we will assume rewards are adversarial, and we will state explicitly other assumptions that qualify a bound or theorem, and specify which kind of regret the bound applies to. We note as well that these ideas of bounds are truly approximations for optimal regret. Also, we note that many papers prove their bounds in the limit as T approaches infinity, whereas much of the work we discuss here holds for finite T.

We note that it is impossible to achieve sublinear regret with any deterministic strategy in the non-stochastic model, since an adversary can simply give maximal reward to the actions the player did *not* take. However, the key for all bandit algorithms is to choose the action by sampling it from a probability distribution over actions, $p(t)$, which the player updates as she explores the action-reward space; $p(t)$ is fully determined by the previous plays and rewards received from them. This is known as a *mixed strategy*. Treating actions as random variables drives us to formulate regret bounds

with respect to the expected gains or losses of the player [1].

It is worth briefly mentioning the many connections to reinforcement learning (RL) in the artificial intelligence community, as the MAB problem is similar to a Partially Observable Markov Decision Process. In both settings, an agent selects actions at each time step and receives feedback in the form of reward signals. Both settings involve an important exploration versus exploitation trade-off, which states that a learner should balance between collecting new information by exploring new actions to make intelligent decisions in the future, with sticking to the best decision she knows so far. However, in the RL setting, there is a notion of state and state-transition probabilities: in each state the set of actions may change, and performing an action may lead to a new state. Reward in the RL setting often depends on both the state (or history of states) and the action played in that state. The RL problem is quite a bit harder in that there can be uncountably many states and it might not be possible to experience every state-action pair. The goal in the RL setting is to learn a policy that maps states to the best actions that maximize the expected cumulative discounted rewards. A MAB, on the other hand, has no state (or one state), and there has been much work proving that nearly all actions can be explored optimally (for a non-adversarial bandit) [3].

Standard notation that will be used throughout this paper:

$$
\begin{aligned}
K &: \text{The number of slot machines or actions} \\
T &: \text{Time horizon of plays, possibly unknown} \\
x(t) \in [0,1]^K &: \text{Reward vector at time t} \\
x_i(t) &: \text{The ith component of the vector x at time t} \\
i_t \in \{1, 2...K\} &: \text{The user's choice of arm at time t} \\
J &: \{j_1, j_2, \ldots, j_T\} \text{ any sequence of actions} \\
x_{i_t}(t) &: \text{The user's reward from choice i at time t} \\
\mathcal{G}_A = \sum_{i=1}^{T} x_{i_t}(t) &: \text{The user's cumulative reward for all actions taken} \\
p(t) \in k\text{-simplex} &: \text{The user's probability distribution over actions at time t} \\
\ell_t \in [0,1]^K &: \text{The loss vector observed or incurred at time t} \\
\mathcal{F} &: \text{a finite set of strategies, } |\mathcal{F}| = F
\end{aligned}
$$

2. Notions of Regret

Regret is what a player feels when she wishes she had taken a different sequence of actions in the past. But which sequence should she compare to? We discuss several of the most common definitions of regret in the community. We note that there is much literature explaining how to convert an algorithm that minimizes one type of regret into an algorithm that also performs well minimizing a different definition of regret. See [2, 1] for some overviews on these meta-algorithms.

2.1. Pure-Strategy (Weak) Regret

The simplest type of regret is *pure strategy* where a player compares against choosing the same action every time[1]. The gain of the *best* pure strategy is the cumulative reward of playing that action for all T time steps which maximizes the reward,

$$\mathcal{G}_{\text{best pure strategy}}(T) = \max_{j \in [K]} \sum_{t=1}^{T} x_j(t)$$

and the regret is the difference in cumulative rewards (or losses) between the best pure strategy and the sequence of actions $\{i_t\}_{t=1}^{T}$ the player chose:

$$\mathcal{R}_{\text{best pure strategy}}(T) = \max_{j \in [K]} \sum_{t=1}^{T} x_j(t) - \sum_{t=1}^{T} x_{i_t}(t)$$

Since the action i_t played is randomized, it makes more sense to analyze the expected regret:

$$\mathbb{E}[\mathcal{R}_{\text{best pure strategy}}(T)] = \max_{j \in [K]} \sum_{t=1}^{T} x_j(t) - \mathbb{E}[\mathcal{G}_A]$$

The authors of [5] also proved *high probability* bounds on the expected pure strategy regret of the form

$$P[\mathcal{G}_{\text{best pure strategy}}(T) > \mathcal{G}_A(T) + \epsilon] \leq \delta$$

[1] Confusingly, this is also called weak regret in some literature, and various forms of this are called internal or external regret in [4]

specifically, this was $O\big(\sqrt{KTln(KT/\delta)}\big)$ with probability at least $1-\delta$ uniformly over T with adversarial rewards.

Lai and Robbins proved that for MAB in the *stochastic* rewards setting, the player can achieve $\mathcal{R}_{\text{best pure strategy}}(T) = O(ln(T))$ for the limit as T approaches infinity; this is also the optimal bound under those assumptions [6]. Thus, providing bounds that hold uniformly over finite T leads to a substantial gap in guarantees.

2.2. Worst Case Regret

Auer *et al.* describe regret in the adversarial setting with respect to the best possible sequence of actions that could have been played (giving rise to the worst possible regret) [5]. For any sequence of actions, $(j_1, j_2, \ldots, j_T)$ [2], the *worse-case* regret is the difference between the the cumulative reward of this sequence, $\mathcal{G}_{(j_1,j_2,\ldots,j_T)}$, and the sequence of actions the player chose:

$$\mathcal{R}_{\text{worst-case}}(T) = \sum_{t=1}^{T} x_{j_t}(t) - \sum_{t=1}^{T} x_{i_t}(t) = \mathcal{G}_{(j_1,j_2,\ldots,j_T)} - \mathcal{G}_A$$

Obviously the worst case regret cannot vanish if we consider all K^T possible sequences with respect to the most adversarial assignment of rewards. To work around this issue, a notion of *hardness*, H, of a sequence is defined as

$$H(\{j_1, j_2, \ldots, j_T\}) = 1 + |\{t \in [T-1] : j_t \neq j_{t+1}\}|$$

the number of "switches" between actions. For instance, $H(\{1,1,2,3\}) = 3$. The worst-case regret bounds intuitively grow with the hardness of the sequence to which the player's actions are compared. The bounds of worst-case regret are respective of the expected reward of the player:

$$\mathbb{E}[\mathcal{R}_{\text{worst-case}}(T)] = \mathcal{G}_{(j_1,j_2,\ldots,j_T)} - \mathbb{E}[\mathcal{G}_A]$$

The most general worst-case regret bound for the adversarial setting is $O\big(\sqrt{SKTln(KT)}\big)$ [5], which holds uniformly over T and simultaneously for all assignments of rewards to all sequences $J = \{j_t\}_{t=1}^{T}$ such that $H(J) \leq S$.

[2] particularly, the sequence that incurs the greatest gain

2.3. External Regret

Another useful notion of regret compares the strategy the player employs to a fixed set of strategies[3] $\mathcal{F}$, where a strategy $f \in \mathcal{F}$ is a function[4] that returns a probability distribution vector over the actions at every time step, $\mathbf{p^f(t)}$. A common instance in gameplay is a player asking an expert whether x is the right action to take in time t, and the expert will either affirm her decision or recommend an alternative (both in the form of a probability distribution). Given assignments of rewards to each of the F strategies, the expected gain of the best strategy is

$$\mathcal{G}_{\text{best-strategy}}(T) = \max_{f \in \mathcal{F}} \sum_{t=1}^{T} \mathbf{p^f(t)} \cdot \mathbf{x(t)}$$

where $\mathbf{p^f(t)} \cdot \mathbf{x(t)}$ is the expected gain of strategy f at time t. This leads to a definition of regret for the best strategy:

$$\mathcal{R}_{\text{external}}(T) = \mathcal{G}_{\text{best-strategy}}(T) - \mathcal{G}_A(T)$$

Interpreting the F strategies as "experts", which we use interchangeably, in the setting of prediction from expert advice, the goal of the player is combine the experts' advice so that the expected regret is no worse than that of the best expert. The expected regret with respect to the best strategy can be bounded using algorithm **Exp4** uniformly over T with high probability as

$$\mathcal{R}(\text{best-strategy}) = O\big(\sqrt{TKln(F)}\big)$$

where the F can even be exponential. This can be improved, in many cases quite substantially, to $O(\sqrt{gKln(F)})$ if it is known that $\mathcal{G}_{\text{best-strategy}}(T) \leq g$ [5]

Note that there are algorithms (like **Exp4**) that do not require the full vector of returns $\mathbf{x(t)}$ and can provide bounds on the expected regret just with partial information.

[3]Several authors define external regret as the best pure strategy regret. We take the approach of (Auer *et al.* 2002) and generalize it.

[4]There are no assumptions or restrictions over the inputs or how these functions generate advice

2.3.1. Other Terminology and Special Cases

Some authors define *external regret* as the expected regret of the single best action played every time, which is what we call best pure-strategy regret [1, 2].

Separately, one notion of *internal regret* captures whether the player should have taken action u at every time step where she took action v, for all $K(K-1)$ pairs of (u, v). This is also a special case of our definition of external regret: consider $K(K-1)$ experts $f_{(u,v)}$, $u \neq v$ such that $f_{(u,v)}(x) = v$ if $x = u$ and x otherwise. That is, when the player asks whether she should play action x, $f_{(u,v)}$ will affirm her decision as long as $x \neq u$, else $f_{(u,v)}$ will recommend v instead. Technically $f_{(u,v)}$ would return a probability vector that is one-hot on the dimension of the recommendation and zero everywhere else. Using these two definitions, it can be shown that internal regret here upper bounds the external regret, but the converse is not true [1].

Generalizing to the extreme, consider K^K experts representing all functions $f : [K] \to [K]$, and bound this *swap regret* by $O\big(\sqrt{TKln(K)}\big)$ in the full-information setting. Both internal and external regret using these definitions are bounded above by swap regret, but the swap regret is itself bounded above by K times the internal regret [2, 1]. In the arguably hardest scenario with a non-oblivious adversary and partial-information feedback, regret is lower bounded by $\Omega(\sqrt{TK})$ [2].

2.4. Wide-Range Regret and Sleeping Experts

The pool of strategies or expert setting is clearly most general, and some researchers allow flexibility for experts to be "asleep" and give no predictions during a subset of time steps (this is often modeled as an activation function $A(f, t) \in [0, 1]$ which states how active or confident expert f is at time step t, with 1 being fully active and 0 being abstain). The results generally show that for each expert, the loss a player suffered is not much more than the loss of the expert while he was awake [7, 2, 8]

With arbitrary classes of experts and boolean activation functions, Lehrer proved sublinear regret can be achieved *in the limit* as T approaches infinity, a result which was improved to hold for finite T and real valued activations by Blum and Mansour [9, 2]

3. Origins of Multi-Armed Bandit

Research on MABs begins in the fields of statistics from before 1950. The original problem is given two populations, for example items from two different factories, with some fixed distributions, we would like to compare these populations for quality control purposes. One example would be to find the difference in the mean of some measurement from these populations. However, performing the measurements on a sample of the populations is expensive, so the task is how best to distribute these observations between the two populations [10]. Much of the formalism for what is called sequential design, updating how one performs their experiments based on what they have already seen, stems from Abraham Wald [11]. It is not too hard to see this problem is more or less the same as the stochastic multi-armed bandit problem: determining the best way to draw from fixed distributions to minimize error.

4. Stochastic Problem

The simplest case of MAB is as follows: rewards for each machine are drawn from some fixed distribution, and the distributions of all the machines are all generated from the same univariate density function. Lai and Robbins[6] first devised an algorithm in 1985 to achieve an $O(\log T)$ best pure strategy regret bound as T approaches to infinity using statistical models. The set of rules developed, transformed into an algorithm, is as follows:

Algorithm 1: Asymptotically Efficient Adaptive Allocation Rules

Parameter(s): $0 < \delta < 1/K$
for $t = 1...T$ **do**
  if $t \leq K$ **then**
    choose action $i = t$
  else
    Let action i be $i \in [1..K]$ that maximizes the estimated mean $\hat{\mu_t}(i)$ such that action i has been choose at least δt times.
    Take action i if $\hat{\mu_t}(i)$ does not fall within an upper confidence bound of another $j \in [1..K]$. Otherwise take action j.
  end
end

The basic idea for this set of rules is that you only explore if you are not confident enough that you are choosing from the best distribution. Since

these rules are asymptotic, as the number of actions goes to infinity we are eventually confident that we are picking from the best distribution. The stochastic version is interesting to examine, but we will focus more on the non-stochastic problem.

5. Non-Stochastic Problem

In this section, we will operate under the assumption of non-oblivious, adversarial assignments of rewards, with all rewards $x_{i_t}(t) \in [0, 1]$. Some authors generalize to rewards on the interval $[a, b]$, but it does not change the bounds meaningfully. In all the algorithms below we state the best *pure-strategy regret* unless otherwise stated. In this non-stochastic setting, it has been shown that the lower bound on expected regret is $\Omega(\sqrt{KT})$ [5]. For the state of the art algorithms there is still a roughly logarithmic factor gap above this lower bound, but these are the best known algorithms for each setting. It remains an open problem to close this gap in many cases.

5.1. Hedge and Full Information Settings

Under the assumption of full information, it becomes easier to identify what the best choices are, and we can weight the actions based on how well they have performed in the past. The Hedge algorithm keeps an internal vector of counters storing rewards obtained by each action $\mathbf{s}$; action i is chosen at time t with weight proportional to $(1+\alpha)^{s_i(t)}$, which can be renormalized at every time step to $\mathbf{p(t)}$. Using a variation on weighted majority algorithm adapted for the full-information yet adversarial bandit setting, the Hedge algorithm [12] reads:

Algorithm 2: Hedge

Parameter(s): $\alpha > 0$
Let $\boldsymbol{s}$ = Vector of total rewards up to current time t
Initialize: $s_i(1) \leftarrow 0 \forall i \in [1...K]$
for $t = 1...T$ **do**
  Choose action i_t according to the distribution $p(t)$:
    $p_i(t) = \frac{(1+\alpha)^{s_i(t)}}{\sum_{j=1}^{K}(1+\alpha)^{s_j(t)}}$
  Receive reward vector $x(t)$ and reward for choice i, $x_{i_t}(t)$
  Set $s_i(t+1) \leftarrow s_i(t) + x_i(t)\ \forall\ i \in [1...K]$
end

Using **Hedge**, we can bound the regret via:

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

$$\mathbb{E}[\mathcal{G}_{Hedge}] \geq \frac{\mathcal{G}_{Best} \ln(1+\alpha) - \ln(K)}{\alpha} \tag{2}$$

$$\geq \mathcal{G}_{Best} - \frac{\alpha}{2}\mathcal{G}_{Best} - \frac{\ln(K)}{\alpha} \tag{3}$$

$$\mathcal{R}_{Hedge} \leq \sqrt{2T \ln(K)} \tag{4}$$

Equation (1) comes from Freund and Schapire's work on bounding the loss of the Hedge algorithm in the context of boosting problems [8]. Given a proper setting of α, we arrive at the bound in (4). Thus, the **Hedge** algorithm gives us a $O(\sqrt{T \ln(K)})$ bound on regret [12].

5.2. EXP3 and Partial Information Settings

In more realistic scenarios, we are unlikely to obtain the full rewards vector $\mathbf{x(t)}$ because only events that actually occurred can be measured. We turn now to the partial information scenario and build off of the Hedge algorithm previously described. Instead of tracking all rewards for each arm, **Exp3** tracks those which were received, and constructs a *simulated reward* vector $\hat{x}$. Additionally, **Exp3** incorporates a uniform prior over actions so that all actions encouraged to be played at some point. Algorithm **Exp3** stands for "Exponential-weight algorithm for Exploration and Exploitation" [12]:

Algorithm 3: Exp3

Parameter(s): $\alpha > 0, \gamma \in [0, 1]$
Initialize: As in **Hedge**
for $t = 1...T$ **do**
  Get the distribution $p(t)$ from **Hedge**
  Select action i_t to be j with probability:
    $\hat{p}_j(t) = (1-\gamma)p_j(t) + \frac{\gamma}{K}$
  Receive reward $x_{i_t}(t)$
  Update simulated reward vector as:
    $\hat{x}_j(t) = \frac{\gamma}{K}\frac{x_{i_t}(t)}{\hat{p}_{i_t}(t)}$ if $j = i_t$, 0 otherwise $\forall\ j \in [1...K]$
  Pass simulated reward vector $\hat{x}$ to **Hedge**
end

Using **Exp3**, we can bound the best pure-strategy regret:

$$\mathbb{E}[\mathcal{G}_{Exp3}] \geq \frac{1-\gamma}{\alpha}(\mathcal{G}_{Best} \ln(1+\alpha) - \frac{K \ln(K)}{\gamma}) \tag{1}$$

$$\mathcal{R}_{Exp3} \leq \frac{3}{\sqrt[3]{2}} g^{2/3} (K \ln(K))^{1/3} \tag{2}$$

Where (1) comes from first identifying a lower bound on the total reward, and then applying the modifications to our $\hat{x}$ vector to the **Hedge** algorithm proof. (2) comes from the assumption that $\mathcal{G}_{\text{best-strategy}} \leq g$, (we shorten $\mathcal{G}_{\text{best-strategy}}$ to $\mathcal{G}_{Best}$), $\alpha = \sqrt[3]{(4K \ln(K))/g}$, and γ is the $\min\{1, \sqrt[3]{(K \ln(K))/(2g)}\}$. Thus, the **Exp3** algorithm gives us a $O(g^{2/3}(K \ln(K))^{1/3})$ bound on regret [12].

Auer *et al.* later improved upon this bound by using an exponential instead of multiplicative update, where final step to update $s(t)$ (in the **Hedge** algorithm) is actually $s_i(t+1) = s_i(t) \exp(\hat{x}_i(t))$ [5]. Additionally, the initialization step sets all values in s to be 1 rather than 0. These modifications will become standard in the remaining algorithms we present, so we will denote them as *"exponential updates"* henceforth. The pure-strategy regret bounds on exponential updates is $O(\sqrt{gK \ln(K)})$.

5.3. EXP3.1 and Unbounded Gain

Notice both bounds for **Exp3** require knowledge apriori of a $g \geq \mathcal{G}_{Best}$, in order to properly choose α and γ. This is often not realistic in all scenarios, and **Exp3.1** [12] drops this assumption. Using **Exp3** as a subroutine, **Exp3.1** accumulates rewards and doubles its estimate of $\mathcal{G}_{Best}$ whenever the reward for one action becomes bigger than the current estimate of $\mathcal{G}_{Best}$; it subsequently restarts the **Exp3** subroutine with new and improved parameters.

Algorithm 4: Exp3.1

Initialize: $n \leftarrow 1, \bar{s}_i \leftarrow 0 \; \forall \; i \in [1...K]$

1 Set $g(n) \leftarrow 2^n$

2 Restart **Exp3** using the parameters:

$g = g(n), \alpha = \sqrt[3]{(4K \ln(K))/g}, \gamma = \min(1, \sqrt[3]{(K \ln(K))/(2g)})$

3 Let **Exp3** choose action i_t, updating the rewards as:

$\bar{s}_{i_t} \leftarrow \bar{s}_{i_t} + \frac{x_{i_t}(t)}{\hat{p}_{i_t}(t)}$

4 **if** $\max_i \bar{s}_i > g(n) - \frac{K}{\gamma}$ **then**

5 set $n \leftarrow n + 1$

6 Go to Step 1.

else

7 Go to Step 3.

end

The best pure-strategy regret bound of **Exp3.1** is:

$$\mathcal{R}_{pure-strategy}[Exp3.1] \leq 43\big(\mathcal{G}_{Best}^{2/3} \sqrt[3]{K \ln(K)} + K^2 \sqrt[3]{\ln(K)}\big) \tag{1}$$

Where the above is derived by identifying a bound on the regret from each round, coupled with a bound on the number of rounds **Exp3.1** goes through before identifying a proper $\mathcal{G}_{Best}$. If $\mathcal{G}_{Best} = \Omega(K^3)$, then **Exp3.1** bounds the regret [12] to $O(\mathcal{G}_{Best}^{2/3}(K \ln(K))^{1/3})$ for all. Just like in **Exp3**, the authors later revised the algorithm to use exponential updates, which achieve the improved regret bounds of $O(\sqrt{K \mathcal{G}_{Best} \ln(K)})$ [5].

So far, we have only used the notion of pure-strategy regret. However, this compares the player's action sequence to a very limited set of sequences, and it would be useful to compare it to all possible sequences, the *worst case* regret, which involves the definition of the *hardness* H of a sequence introduced previously. Algorithm **Exp3.S** modifies **Exp3** to achieve good performance on worst-case regret [5].

Algorithm 5: Exp3.S

Parameter(s): $\alpha > 0, \gamma \in [0, 1]$
Initialize: $s_i(1) \leftarrow 1 \forall i \in [1...K]$
for $t = 1...T$ **do**
  Select action i_t according to the distribution $p(t)$:
    $p_i(t) = (1-\gamma)\frac{s_i(t)}{\sum_{j=1}^{K} s_j(t)} + \frac{\gamma}{K}\ \forall i = 1...K$
  Receive reward $x_{i_t}(t)$
  Set $\hat{x}_j(t) = x_j(t)/p_j(t)$ if $j = i_t$, 0 otherwise
  Update $s(t)$ as:
    $s_j(t+1) = s_j(t)\exp\left(\gamma\hat{x}_j(t)/K\right) + \frac{e\alpha}{K}\sum_{i=1}^{K} s_i(t)$
end

Auer *et al.* bound the worst-case regret as

$$\mathbb{E}[\mathcal{G}_{Exp3.S}] \geq \max_J(\mathcal{G}_J - H(J)\sqrt{KT\ln(KT)} - 2e\sqrt{\frac{KT}{\ln(KT)}}) \tag{1}$$

$$\mathcal{R}_{Exp3.S} \leq H(J)\sqrt{KT\ln(KT)} + 2e\sqrt{\frac{KT}{\ln(KT)}} \tag{2}$$

Where $\mathcal{G}_J$ represents the gains for any sequence of pulls, J, whose hardness is $H(J)$. The regret bounds can be rewritten as $O(H(J)\sqrt{KT\ln(KT)})$ [5].

5.4. Expert Advice and Strategy Regret

Suppose now the player has access to a pool of experts or strategies from $\mathcal{F}$, each of which recommend a probability distribution over actions. In this scenario, the goal is no longer to minimize pure-strategy regret (which is a special case if $\mathcal{F}$ includes the pure strategies), but instead, minimize regret with respect to the rewards received had the player chosen one of the expert's strategies for the entire game. This is known as *external* or strategy regret. We denote the probability distribution that expert $f \in \mathcal{F}$ recommends at time t as $p^f(t) \in [0, 1]^K$, $\sum_{i=1}^{K} p_i^f(t) = 1$. In this setting, the best expert $\mathcal{G}_{Best}$ is the one that would have returned the greatest reward had his sequences been played:

$$\mathcal{G}_{Best} = \max_{1 \le l \le F} E_{i_1 \ldots i_T} \big[\sum_{t=1}^{T} p^f(t) \cdot x(t) \big]$$
$$\mathbb{E}[\mathcal{R}_{external}] = \mathcal{G}_{Best} - \mathbb{E}[\mathcal{G}_A]$$

Adhering to the partial information assumption, **Exp4** [12] (Exponential-weight algorithm for Exploration and Exploitation using Expert advice) modifies **Exp3** to generate a new strategy from a weighted combination of expert strategies that minimize external regret [12]. **Exp4** enlists the **Hedge** algorithm as a subroutine.

Algorithm 6: Exp4

Parameter(s): $\alpha > 0, \gamma \in [0, 1]$
Initialize: As in **Hedge**, but replace K with F
for *t = 1...T* **do**
  Get the distribution $q(t)$ from **Hedge**
  Get the advice vectors $p^f(t)\ \forall f \in \mathcal{F}$; define the current $p(t)$:
    $p(t) = \sum_{k=1}^{F} q_k(t) p^f(t)$
  Select action i_t to be j with probability:
    $\hat{p}_j(t) = (1 - \gamma) p_j(t) + \gamma / K$
  Receive reward $x_{i_t}(t)$
  Compute simulated reward vector as:
    $\hat{x}_j(t) = \frac{\gamma}{K} \cdot \frac{x_{i_t}(t)}{\hat{p}_{i_t}(t)}$ if $j = i_t$, 0 otherwise $\ \forall\ j \in [1...K]$
  Define vector $y(t)$ to be:
    $y_j(t) = p^f(t) \hat{x}(t)$
  Feed vector $y(t)$ to **Hedge**
end

The *external regret* is bounded above:

$$\mathbb{E}[\mathcal{G}_{Exp4}] \ge \mathcal{G}_{Best} - (\gamma + \frac{\alpha}{2}) \mathcal{G}_{Best} - \frac{K \ln(F)}{\alpha \gamma} \tag{1}$$
$$\mathcal{R}_{external}[Exp4] = O(g^{2/3} (K \ln(F))^{1/3}) \tag{2}$$

The analysis compares the expected gain of each action to that of the theoretical best expert. This analysis is a slight modification on **Exp3**, to take into account a F dimensional vector rather than a K dimensional vector. The $O(g^{2/3}(K \ln(F))^{1/3})$ bound can improved to $O(\sqrt{gK \ln(F)})$ using exponential updates [5].

6. Conclusion

To summarize the results in the literature, we include the following table of regret bounds, omitting external and swap regrets, for which there are a few results.

Table 1: Algorithms and Their Regret Bounds

Algorithm	**Pure Strategy**	**Worst Case**
Hedge (*full information*)	$O(\sqrt{T \ln(K)})$	N/A
Exp3 (*partial information, known* G_{Best})	$O(\sqrt{gK \ln(K)})$	N/A
Exp3.1 (*partial information*)	$O(\sqrt{gK \ln(K)})$	N/A
Exp3.S (*partial information*)	N/A	$O(H(J)\sqrt{KT \ln(KT)})$
Exp4 (*partial information, advice of many experts*)	$O(\sqrt{gK \ln(F)})$	N/A

Here we present a summary of the regret bounds achieved by various algorithms, and the assumptions under which those algorithms operate are italicized (all of them receive non-stochastic or adversarial rewards, and the bounds hold for any finite T). The rewards at each step after playing one of K actions are on $[0, 1]$; if g, an upper bound to the cumulative reward over time, is not known, note that $g = O(T)$ always. Bounds that were not proved are marked N/A.

More recent work in dealing with MABs have generally changed the problem in other ways from the assumptions made in the presented algorithms. For example, there is the idea of contextual bandits, or using some additional context to make decisions at each time t. However, we believe that this look at non-stochastic and brief stochastic versions of the MAB problem are still important to understand how many variants and bounds there are on this problem.

[1] N. Cesa-Bianchi and G. Lugosi, *Prediction, learning, and games*. Cambridge university press, 2006.

[2] A. Blum and Y. Mansour, "From external to internal regret," *Journal of Machine Learning Research*, vol. 8, no. Jun, pp. 1307–1324, 2007.

[3] S. Russell, P. Norvig, and A. Intelligence, "A modern approach," *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, vol. 25, p. 27, 1995.

[4] R. Zheng and C. Hua, "Sequential learning and decision-making in wireless resource management," 2017.

[5] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The nonstochastic multiarmed bandit problem," *SIAM journal on computing*, vol. 32, no. 1, pp. 48–77, 2002.

[6] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances in applied mathematics*, vol. 6, no. 1, pp. 4–22, 1985.

[7] A. Blum, "Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain," *Machine Learning*, vol. 26, no. 1, pp. 5–23, 1997.

[8] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *European conference on computational learning theory*, pp. 23–37, Springer, 1995.

[9] E. Lehrer, "A wide range no-regret theorem," *Games and Economic Behavior*, vol. 42, no. 1, pp. 101–115, 2003.

[10] H. Robbins, "Some aspects of the sequential design of experiments," *Bulletin of the American Mathematical Society*, vol. 58, no. 5, pp. 527–535, 1952.

[11] A. Wald, "Statistical decision functions," *The Annals of Mathematical Statistics*, pp. 165–205, 1949.

[12] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "Gambling in a rigged casino: The adversarial multi-armed bandit problem," in *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pp. 322–331, IEEE, 1995.