From External to Internal Regret

Avrim Blum[1] * and Yishay Mansour[2] **

[1] School of Computer Science,
Carnegie Mellon University, Pittsburgh, PA 15213. avrim@cs.cmu.edu
[2] School of Computer Science, Tel-Aviv University, ISRAEL. mansour@cs.tau.ac.il

Abstract. External regret compares the performance of an online algorithm, selecting among N actions, to the performance of the best of those actions in hindsight. Internal regret compares the loss of an online algorithm to the loss of a modified online algorithm, which consistently replaces one action by another.

In this paper, we give a simple generic reduction that, given an algorithm for the external regret problem, converts it to an efficient online algorithm for the internal regret problem. We provide methods that work both in the *full information* model, in which the loss of every action is observed at each time step, and the *partial information* (bandit) model, where at each time step only the loss of the selected action is observed. The importance of internal regret in game theory is due to the fact that in a general game, if each player has sublinear internal regret, then the empirical frequencies converge to a correlated equilibrium.

For external regret we also derive a quantitative regret bound for a very general setting of regret, which includes an arbitrary set of modification rules (that possibly modify the online algorithm) and an arbitrary set of time selection functions (each giving different weight to each time step). The regret for a given time selection and modification rule is the difference between the cost of the online algorithm and the cost of the modified online algorithm, where the costs are weighted by the time selection function. This can be viewed as a generalization of the previously-studied *sleeping experts* setting.

1 Introduction

The motivation behind regret analysis might be viewed as the following: we design a sophisticated online algorithm that deals with various issues of uncertainty and decision making, and sell it to a client. Our online algorithm runs for some time and incurs a certain loss. We would like to avoid the embarrassment that

* This work was supported in part by NSF grants CCR-0105488 and IIS-0312814.

** The work was done while the author was a fellow in the Institute of Advance studies, Hebrew University. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778, by a grant no. 1079/04 from the Israel Science Foundation and an IBM faculty award. This publication only reflects the authors' views.

our client will come back to us and claim that in *retrospect* we could have incurred a much lower loss if we used his simple alternative policy π. The regret of our online algorithm is the difference between the loss of our algorithm and the loss using π. Different notions of regret quantify differently what is considered to be a "simple" alternative policy.

At a high level one can split alternative policies into two categories. The first consists of alternative policies that are independent from the online algorithm's action selection, as is done in *external regret*. External regret, also called the *best expert* problem, compares the online algorithm's cost to the best of N actions in retrospect [19, 15, 24, 17, 18, 6]. This implies that the simple alternative policy performs the same action in all time steps, which indeed is quite simple. Nonetheless, one important application of external regret to online algorithm analysis is a general methodology of developing online algorithms whose performance matches that of an optimal static offline algorithm by modeling the possible static solutions as different actions.

The second category are those alternative policies that consider the online sequence of actions and suggest a simple modification to it, such as "every time you bought IBM, you should have bought Microsoft instead." This notion is captured by *internal regret* [13]. Specifically, internal regret allows one to modify the online action sequence by changing every occurrence of a given action i by an alternative action j. Specific low internal regret algorithms were derived in [20, 12–14, 7], where the use of the approachability theorem [4] has played an important role in some of the algorithms [20, 12, 14].

One of the main contributions of our work is to show a simple online way to efficiently convert any external regret algorithm into an *internal* regret algorithm. Our guarantee is somewhat stronger than internal regret and we call it *swap regret*, which allows one to *simultaneously* swap multiple pairs of actions. (If there are N actions total, then swap-regret is bounded by N times the internal regret.) Using known results for external regret we can derive a swap regret bound of $O(N\sqrt{T \log N} + N \log N)$, and with additional optimization we are able to reduce this regret bound to $O(\sqrt{NT \log N} + N \log N \log T)$. We also show an $\Omega(\sqrt{NT})$ lower bound for the case of randomized online algorithms against an adaptive adversary.

The importance of internal regret is due to its tight connection to correlated equilibria [3]. For a general-sum game of any finite number of players, a distribution Q over the joint action space is a correlated equilibrium if every player would have zero internal regret when playing it. In a repeated game scenario, if each player uses an action selection algorithm whose internal regret is sublinear in T, then the empirical distribution of the players actions converges to a correlated equilibrium (see, e.g. [20]). In fact, we point out that the deviation from a correlated equilibrium is bounded exactly by the average swap regret of the players.

We also extend our internal regret results to the *partial information model*, also called the *adversarial multi-armed bandit* (MAB) problem [2]. In this model, the online algorithm only gets to observe the loss of the action actually selected, and does not see the losses of the actions not chosen. For example, if you are driving in rush-hour traffic and need to select which of several routes to take, you only observe the travel time on the route actually taken. If we view this as an online problem, each day selecting which route to take on that day, then this fits the MAB setting. Furthermore, the route-choosing problem can be viewed as a general-sum game: your travel time depends on the choices of the other drivers as well. Thus, if *every* driver uses a low internal-regret algorithm, then traffic patterns will converge to a correlated equilibrium. For the MAB problem, our combining algorithm requires additional assumptions on the base external-regret MAB algorithm: a smoothness in behavior when the actions played are taken from a somewhat different distribution than the one proposed by the algorithm. Luckily, these conditions are satisfied by existing external-regret MAB algorithms such as that of Auer et al. [2]. For the multi-armed bandit setting, we derive an $O(\sqrt{N^3 T \log N} + N^2 \log N)$ swap-regret bound. Thus, after $T = O(\frac{1}{\epsilon^2} N^3 \log N)$ rounds, the empirical distribution on the history is an ϵ-correlated equlibrium. (The work of [21] also gives a multi-armed bandit algorithm whose internal regret is sublinear in T, but does not derive explicit bounds.)

One can also envision broader classes of regret. Lehrer [23] defines a notion of *wide range regret* that allows for arbitrary action-modification rules, which might depend on history, and also Boolean time selection functions (which determine which subset of times is relevant). Using the approachability theorem [4], he shows a scheme that in the limit achieves no regret (regret is sublinear in T). While [23] derives the regret bounds in the limit, we derive finite-time regret bounds for this setting. We show that for any family of N actions, M time selection functions and K modification rules, the maximum regret with respect to any selection function and modification rule is bounded by $O(\sqrt{TN \log(MK)} + N \log(MK))$. Our model also handles the case where the time selection functions are not Boolean, but rather reals in $[0, 1]$.

This latter result can be viewed as a generalization of the *sleeping experts* setting of [5, 16]. In the sleeping experts problem, we again have a set of experts, but on any given time step, each expert may be awake (making a prediction) or asleep (not predicting). This is a natural model for combining a collection of if-then rules that only make predictions when the "if" portion of the rule is satisfied, and this setting has had application in domains ranging from managing a calendar [5] to text-categorization [11] to learning how to formulate web search-engine queries [10]. By converting each such sleeping-expert into a pair $\langle$expert, time-selection function$\rangle$, we achieve the desired guarantee that for each sleeping-expert, our loss *during the time that expert was awake* is not much more than its loss in that period. Moreover, by using non-Boolean time-selection functions, we can naturally handle prediction rules that have varying degrees of confidence in their predictions and achieve a confidence-weighted notion of regret.

We also study the case of deterministic Boolean prediction in the setting of time selection functions. We derive a deterministic online algorithm whose number of weighted errors, with respect to any time selection function from our class of M selection functions is at most $3OPT + 1 + 2\log M$, where OPT is the best constant prediction for that time selection function. (For lack of space, the proof is omitted in this extended abstract.)

Recent related work. It was brought to our attention [25] that comparable results can be achieved based on independent work appearing in the journal version of [26]: specifically, the results regarding the relation between external and internal regret [27] and the multi-armed bandit setting [8]. In comparison to [27], we are able to achieve a better swap regret guarantee in polynomial time (a straightforward application of [27] to swap regret would require time-complexity $\Omega(N^N)$; alternatively, they can achieve a good internal-regret bound in polynomial time, but then their swap regret bound becomes worse by a factor of $\sqrt{N}$). On the other hand, work of [27] is applicable to a wider range of loss functions, which also capture scenarios arising in portfolio selection. We should stress that the above techniques are very different from the techniques proposed in our work.

2 Model and Preliminaries

We assume an adversarial online model where there are N available actions $\{1, \ldots, N\}$. At each time step t, an online algorithm H selects a distribution p^t over the N actions. After that, the adversary selects a loss vector $\ell^t \in [0,1]^N$, where $\ell_i^t \in [0,1]$ is the loss of the i-th action at time t. In the *full information model*, the online algorithm receives the loss vector ℓ^t and experiences a loss $\ell_H^t = \sum_{i=1}^N p_i^t \ell_i^t$. In the *partial information model*, the online algorithms receives $(\ell_{k^t}^t, k^t)$, where k^t is distributed according to p^t, and $\ell_H^t = \ell_{k^t}^t$ is its loss. The loss of the i-th action during the first T time steps is $L_i^T = \sum_{t=1}^T \ell_i^t$, and the loss of H is $L_H^T = \sum_{t=1}^T \ell_H^t$. The aim for the external regret setting is to design an online algorithm that will be able to approach the best action, namely, to have a loss close to $L_{min}^T = \min_i L_i^T$. Formally we would like to minimize the external regret $R = L_H^T - L_{min}^T$.

We introduce a notion of a *time selection* function. A time selection function I is a function over the time steps mapping each time step to $[0,1]$. That is, $I : \{1, \ldots, T\} \to [0,1]$. The loss of action j using time-selector I is $L_{j,I}^T = \sum_t I(t) \ell_j^t$. Similarly we define $L_{H,I}$, the loss of the online algorithm H with respect to time selection function I, as $L_{H,I}^T = \sum_t I(t) \ell_H^t$, where ℓ_H^t is the loss of H at time t. This notion of experts with time selection is very similar to the notion of "sleeping experts" studied in [16]. Specifically, for each action j and time selection function I, one can view the pair (j, I) as an expert that is "awake" when $I(t) = 1$ and "asleep" when $I(t) = 0$ (and perhaps "partially awake" when $I(t) \in (0,1)$).

We also consider modification rules that modify the actions selected by the online algorithm, producing an alternative strategy we will want to compete against. A *modification rule* F has as input the history and an action choice and outputs a (possibly different) action. (We denote by F^t the function F at time t, including any dependency on the history.) Given a sequence of probability distributions p^t used by an online algorithm H, and a modification rule F, we define a new sequence of probability distributions $f^t = F^t(p^t)$, where $f_i^t = \sum_{j:F^t(j)=i} p_j^t$. The loss of the modified sequence is $L_{H,F} = \sum_t \sum_i f_i^t \ell_i^t$. Similarly, given a time selection function I and a modification rule F we define $L_{H,I,F} = \sum_t \sum_i I(t) f_i^t \ell_i^t$.

In our setting we assume a finite class of N actions, $\{1, \ldots, N\}$, a finite set $\mathcal{F}$ of K modification rules, and a finite set $\mathcal{I}$ of M time selection function. Given a sequence of loss vectors, the regret of an online algorithm H with respect to the N actions, the K modification rules, and the M time selection functions, is

$$R_H^{\mathcal{I},\mathcal{F}} = \max_{I \in \mathcal{I}} \max_{F \in \mathcal{F}} \{L_{H,I} - L_{H,I,F}\}.$$

Note that the external regret setting is equivalent to having a single time-selection function ($I(t) = 1$ for all t) and a set $\mathcal{F}^{ex}$ of N modification rules F_i, where F_i always outputs action i. For internal regret, the set $\mathcal{F}^{in}$ consists of $N(N-1)$ modification rules $F_{i,j}$, where $F_{i,j}(i) = j$ and $F_{i,j}(i') = i'$ for $i' \neq i$. That is, the internal regret of H is

$$\max_{F \in \mathcal{F}^{in}} \{L_H - L_{H,F}\} \; = \; \max_{i,j} \sum_t p_i^t (\ell_i^t - \ell_j^t).$$

We define a slightly extended class of internal regret which we call *swap regret*. This case has $\mathcal{F}^{sw}$ include all N^N functions $F : \{1, \ldots, N\} \rightarrow \{1, \ldots, N\}$, where the function F swaps the current online action i with $F(i)$ (which can be the same or a different action).

A few simple relationships between the different types of regrets: since $\mathcal{F}^{ex} \subseteq \mathcal{F}^{sw}$ and $\mathcal{F}^{in} \subseteq \mathcal{F}^{sw}$, both external and internal regret are upper-bounded by swap-regret. Also, swap-regret is at most N times larger than internal regret. On the other hand, even with $N = 3$, there are simple examples which separate internal and external regret [26].

2.1 Correlated Equilibria and Swap Regret

We briefly sketch the relationship between correlated equilibria [3] and swap regret.

Definition 1. *A general-sum game $\langle M, (A_i), (s_i) \rangle$ has a finite set M of m players. Player i has a set A_i of N actions and a loss function $s_i : A_i \times (\times_{j \neq i} A_j) \rightarrow [0, 1]$ that maps the action of player i and the actions of the other players to a real number. (We have scaled losses to $[0, 1]$)*

The aim of each player is to minimize its loss. A correlated equilibrium [3] is a distribution P over the joint action space with the following property. Imagine a correlating device draws a vector of actions $\boldsymbol{a}$ using distribution P over $\times A_i$, and gives player i the action a_i from $\boldsymbol{a}$. (Player i is not given any other information regarding $\boldsymbol{a}$.) The probability distribution P is a correlated equilibria if for each player it is its best response to play the suggested action (provided that the other players do not deviate).

We now define an ϵ-correlated equilibrium.

Definition 2. *A joint probability distribution P over $\times A_i$ is an ϵ-correlated equilibria if for every player j and for any function $F : A_j \rightarrow A_j$, we have $E_{a\sim P}[s_j(a_j, a^{-j})] \leq E_{a\sim P}[s_j(F(a_j), a^{-j})] + \epsilon$, where a^{-j} denotes the joint actions of the other players.*

The following theorem relates the empirical distribution of the actions performed by each player, their swap regret and the distance from a correlated equilibrium (see also, [12, 13, 20]).

Theorem 1. *Let $G =< M, (A_i), (s_i) >$ be a game and assume that for T time steps each player follows a strategy that has swap regret of at most $R(T, N)$. The empirical distribution Q of the joint actions played by the players is an $(R(T, N)/T)$-correlated equilibrium, and the loss of each player equals, by definition, its expected loss on Q.*

The above states that the payoff of each player is its payoff in some approximate correlated equilibrium. In addition, it relates the swap regret to the distance from a correlated equilibria. Note that if the average swap regret vanishes then the procedure converges, in the limit, to a correlated equilibria (see [20, 12, 14]).

3 Generic reduction from external to swap regret

We now give a black-box reduction showing how any algorithm A achieving good external regret can be used as a subroutine to achieve good swap regret as well. The high-level idea is as follows. We will instantiate N copies of the external-regret algorithm. At each time step, these algorithms will each give us a probability vector, which we will combine in a particular way to produce our own probability vector p. When we receive a loss vector ℓ, we will partition it among the N algorithms, giving algorithm A_i a fraction p_i (p_i is our probability mass on action i), so that A_i's belief about the loss of action j is $\sum_t p_i^t \ell_j^t$, and matches the cost we would incur putting i's probability mass on j. In the proof, algorithm A_i will in some sense be responsible for ensuring low regret of the $i \rightarrow j$ variety. The key to making this work is that we will be able to define the p's so that the sum of the losses of the algorithms A_i on their own loss vectors matches our overall true loss.

To be specific, let us formalize what we mean by an external regret algorithm.

Definition 3. *An algorithm A has external regret $R(L_{min}, T, N)$ if for any sequence of T losses ℓ^t such that some action has total loss at most L_{min}, for any action $j \in \{1, \ldots, N\}$ we have*

$$L_A^T = \sum_{t=1}^{T} \ell_A^t \leq \sum_{t=1}^{T} \ell_j^t + R(L_{min}, T, N) = L_j^T + R(L_{min}, T, N)$$

We assume we have N algorithms A_i (which could all be identical or different) such that A_i has external regret $R_i(L_{min}, T, N)$. We combine the N algorithms as follows. At each time step t, each algorithm A_i outputs a distribution q_i^t, where $q_{i,j}^t$ is the fraction it assigns action j. We compute a vector p such that $p_j^t = \sum_i p_i^t q_{i,j}^t$. That is, $p = pQ$, where p is the row-vector of our probabilities and Q is the matrix of $q_{i,j}$. (We can view p as a stationary distribution of the Markov Process defined by Q, and it is well known such a p exists and is efficiently computable.) For intuition into this choice of p, notice that it implies we can consider action selection in two equivalent ways. The first is simply using the distribution p to select action j with probability p_j. The second is to select algorithm A_i with probability p_i and then to use algorithm A_i to select the action (which produces distribution pQ).

When the adversary returns ℓ^t, we return to each A_i the loss vector $p_i \ell^t$. So, algorithm A_i experiences loss $(p_i^t \ell^t) \cdot q_i^t = p_i^t (q_i^t \cdot \ell^t)$.

Now we consider the guarantee that we have for algorithm A_i, namely, for any action j,

$$\sum_{t=1}^{T} p_i^t (q_i^t \cdot \ell^t) \leq \sum_{t=1}^{T} p_i^t \ell_j^t + R_i(L_{min}, T, N) \tag{1}$$

If we sum the losses of the N algorithms at any time t, we get $\sum_i p_i^t (q_i^t \cdot \ell^t) = p^t Q^t \ell^t$, where p^t is the row-vector of our probabilities, Q^t is the matrix of $q_{i,j}^t$, and ℓ^t is viewed as a column-vector. By design of p^t, we have $p^t Q^t = p^t$. So, the sum of the perceived losses of the N algorithms is equal to our actual loss $p^t \ell^t$.

Therefore, summing equation (1) over all N algorithms, the left-hand-side sums to L_H^T and so we have that for any function $F : \{1, \ldots, N\} \to \{1, \ldots, N\}$,

$$L_H^T \leq \sum_{i=1}^{N} \sum_{t=1}^{T} p_i^t \ell_{F(i)}^t + \sum_{i=1}^{N} R_i(L_{min}, T, N).$$

We have therefore proven the following theorem.

Theorem 2. *For any N algorithms A_i with regret R_i, for every function F : $\{1, \ldots, N\} \to \{1, \ldots, N\}$, the above algorithm satisfies*

$$L_H \leq L_{H,F} + \sum_{i=1}^{N} R_i(L_{min}, T, N),$$

i.e., the swap-regret of H is at most $\sum_{i=1}^{N} R_i(L_{min}, T, N)$.

A typical *optimized experts algorithm* [24, 17, 2, 6] will have $R(L_{min}, T, N) = O(\sqrt{L_{min} \log N} + \log N)$. (Alternatively, Corollary 4 can be also used to deduce the above bound.) We can immediately derive the following corollary.

Corollary 1. *Using an optimized experts algorithm as the A_i, for every function $F : \{1, \ldots, N\} \to \{1, \ldots, N\}$, we have that*

$$L_H \leq L_{H,F} + O(N\sqrt{T \log N} + N \log N)$$

We can perform a slightly more refined analysis of the bound by having L^i_{min} be the minimum loss for an action in A_i. Since $\sum_{i=1}^N \sqrt{L^i_{min}} \leq \sum_{i=1}^N L^i_{min}$, and this is bounded by T since we scaled the losses given to algorithm A_i at time t by p^t_i, this implies the worst case regret is $O(\sqrt{TN \log N} + N \log N)$. The only problem is that algorithm A_i needs to "know" the value of L^i_{min} to set its internal parameters correctly. One way to avoid this is to use an adaptive method [1]. We can also avoid this problem using the standard doubling approach of starting with $L_{min} = 1$ and each time our guess is violated, we double the bound and restart the online algorithm. The external regret of such a resetting optimized experts algorithm would be

$$\sum_{j=1}^{log L_{min}} O(\sqrt{2^j \log N} + \log N) = O(\sqrt{L_{min} \log N} + \log L_{min} \log N).$$

Going back to our case of N multiple online algorithms A_i, we derive the following,

Corollary 2. *Using resetting optimized experts algorithms as the A_i, for every function $F : \{1, \ldots, N\} \to \{1, \ldots, N\}$, we have that*

$$L_H \leq L_{H,F} + O(\sqrt{TN \log N} + N \log N \log T)$$

One strength of the above general reduction is it ability to accommodate new regret minimization algorithms. For example, using the algorithm of [9] one can get a more refined regret bound, which depends on the second moment.

3.1 Lower bounds for swap regret

Notice that while good algorithms for the experts problem achieve external regret roughly $O(\sqrt{T \log N})$, our swap-regret bounds are roughly $O(\sqrt{TN \log N})$. Or, to put it another way, for external regret one can achieve regret ϵT by time $T = O(\epsilon^{-2} \log N)$, whereas we need $T = O(\epsilon^{-2} N \log N)$ to achieve swap-regret ϵT (or an ϵ-correlated equilibrium). A natural question is whether this is best possible. We give here a partial answer.

First, one tricky issue is that for a given stochastic adversary, the optimal policy for minimizing loss may *not* be the optimal policy for minimizing swap-regret. For example, consider a process in which losses are generated by an almost unbiased coin, with slight biases so that the optimal policy for minimizing loss uses each action T/N times. Because of the variance of the coin flips, in retrospect, most actions can be swapped with an expected gain of $\Omega(\sqrt{(T \log N)/N})$ each, giving a total swap-regret of $\Omega(\sqrt{TN \log N})$ for this policy. However, a policy that just picks a single fixed action would have swap-regret only $O(\sqrt{T \log N})$ even though its expected loss is higher.

We show a lower bound of $\Omega(\sqrt{TN})$ on swap regret, but in a different model. Specifically, we have defined swap regret with respect to the *distribution* p^t produced by the player, rather that the actual action a_t selected from that distribution. In the case that the adversary is oblivious (does not depend on the player's action selection) then the two models have the same expected regret. However we will consider a *dynamic* adversary, whose choices may depend on the player's action selection in previous rounds. In this setting (dynamic adversary *and* regret defined with respect to the action selected from p^t rather than p^t itself) we derive the following theorem.

Theorem 3. *There exists a dynamic adversary such that for any randomized online algorithm A, the expected swap regret of A is $(1-\lambda)\sqrt{TN}/128$, for $T \geq N$ and $\lambda = NTe^{-cN}$ for some constant $c > 0$.*

Proof. (sketch) The adversary behaves as follows. At each time step t, for any action that has been played less than $8T/N$ times by A, the adversary flips a fair coin to set its loss to 0 or 1 (call these *random-loss* actions). However, once an action has been played $8T/N$ times by A, then its loss is 1 from then on (call these *1-loss* actions). Note that at most $N/8$ actions ever become 1-loss actions.

Now, if A in expectation plays 1-loss actions more than $1/4$ of the time, then A will incur such a high loss that it will even have a large *external* regret. Specifically, A will have an expected loss at least $5T/8$, whereas with high probability there will exist some action of total loss at most $T/2$, and this gap exceeds the bounds of the theorem. On the other hand, if A plays random-loss actions at least $3/4$ of the time, then there must be a large number (at least $N/16$) actions that are played at least $T/(4N)$ times by A. However, in the subset of T_i time-steps that A plays some action i, there is a high probability that some other action has loss only $\frac{1}{2}T_i - \frac{1}{4}\sqrt{T_i}$, even if A were able to choose which actions to make 1-loss actions (and thereby remove from consideration) after the fact. On the other hand, A's expected loss is $\frac{1}{2}T_i$. Thus, A has expected swap-regret at least $(N/16)(\frac{1}{4}\sqrt{T/(4N)}) = \sqrt{TN}/128$. The $(1-\lambda)$ factor is to guarantee that the realized values are close to their expected value, with high probability. □

4 Reducing External to Swap Regret in the Partial information model

In the full information setting the learner gets, at the end of each time step, full information on the costs of all the actions. In the partial information (bandit) model, the learner gets information only about the action that was selected. In some applications this is a more plausible model regarding the information the learner can observe.

The reduction in the partial information model is similar to the one of the full information model, but with a few additional complications. We are given N partial information algorithms A_i. At each time step t, each algorithm A_i gives a distribution q_i^t. Our master online algorithm combines them to some distribution p^t which it uses. Given p^t it receives a feedback, but now this includes information only regarding one action, i.e., it receives $(\ell^t_{k^t}, k^t)$, where k^t is distributed according to p^t. We take this feedback and distribute to each algorithm A_i a feedback (b_i^t, k^t), such that $\sum_i b_i^t = \ell^t_{k^t}$. The main technical difficulty is that now the action selected, k^t, is distributed according to p^t and not q_i^t. (For example, it might be that A_i has $q_{i,j}^t = 0$ but it receives a feedback about action j. From A_i's point of view this is impossible! Or, more generally, A_i might start noticing it seems to have a very bad random-number generator.) For this reason, for the reduction to work we need to make a stronger assumption about the guarantees of the algorithms A_i, which luckily is implicit in the algorithms of [2]. Our main result is summarized in the following theorem.

Theorem 4. *Given a multi-arm bandit algorithm satisfying Lemma 1 below (such as the algorithm of [2]), it can be converted to a master online algorithm Int_MAB, such that for every function $F : \{1, \ldots, N\} \to \{1, \ldots, N\}$, we have that*

$$E_{p^t}[L_{Int\_MAB}] \le E_{p^t}[L_{Int\_MAB,F}] + N \cdot R^{MAB}(T, T, N)$$

where $R^{MAB}(C, T, N) = O(\sqrt{CN \log N} + N \log N)$.

Proof. Since results of [2] are stated in terms of maximizing gain rather then minimizing loss we will switch to this notation, and later derive the loss minimization results.

At each time step t the multi-arm bandit algorithm A_i gives a *selection distribution* q_i^t over actions, and given all the selection distributions we compute an *action distribution* p^t. We would like to keep two sets of gains, one is the *real gain*, denoted by b_i^t, and the other is the gain that the MAB algorithm A_i observes $g_{A_i}^t$. Given the action distribution p^t the adversary selects a vector of real gains b_i^t. Our MAB algorithm Int_MAB receives a single feedback $(b_{k^t}^t, k^t)$ where k^t is a random variable that with probability p_j^t equals j. Given b^t it returns to each A_i the pair $(g_{A_i}^t, k^t)$, where the observed gain $g_{A_i}^t$ is based on b^t, p^t and q_i^t. Note that k^t is distributed according to p^t, which may not equal q_i^t: it is for this reason we need to use an MAB algorithm that satisfies Lemma 1.

In order to specify our MAB algorithm, Int_MAB, we need to specify how it selects the action distribution p^t and the observed gains $g_{A_i}^t$. At each time step t, each algorithm A_i outputs a selection distribution q_i^t, where $q_{i,j}^t$ is the probability it assigns action j. We compute an action distribution p^t such that $p_j^t = \sum_i p_i^t q_{i,j}^t$. That is, $p = pQ$, where p is the row-vector of our probabilities and Q is the matrix of $q_{i,j}$. Given p^t the adversary returns a real gain $(b_{k^t}^t, k^t)$, namely, the real gain is of our algorithm $b_{k^t}^t$. We return to each algorithm A_i an observed gain of $g_{A_i}^t = p_i^t b_{k^t}^t q_{i,k^t} / p_{k^t}^t$. (In general, define $g_{i,j}^t = p_i^t b_j^t q_{i,j}^t / p_j^t$, where $b_j^t = 0$ if $j \ne k^t$.) First, we will show that $\sum_{i=1}^N g_{A_i}^t = b_{k^t}^t$ and that $g_{A_i}^t \in [0, 1]$. From the property of the distribution p^t we have that,

$$\sum_{i=1}^{N} g_{A_i}^t = \sum_{i=1}^{N} \frac{p_i^t b_{k^t}^t q_{i,k^t}}{p_{k^t}^t} = \frac{p_{k^t}^t b_{k^t}^t}{p_{k^t}^t} = b_{k^t}^t .$$

This shows that we distribute our real gain between the algorithms A_i, namely that the sum of the observed gains equals the real gain. In addition, it bounds the observed gain that each algorithm A_i receives. Namely, $0 \le g_{A_i}^t \le b_{k^t}^t \le 1$.

In order to describe the guarantee that each external regret multi-arm bandit algorithm A_i has, for our application, we need the following additional definition. At time t let $X_{i,j}^t$ be a random variable such that $X_{i,j}^t = g_{i,j}^t / q_{i,j}^t = p_i^t b_j^t / p_j^t$ if $j = k^t$ and $X_{i,j}^t = 0$ otherwise. The expectation of $X_{i,j}^t$ is

$$E_{k^t \sim p^t}[X_{i,j}^t] = p_j^t \frac{p_i^t b_j^t}{p_j^t} = p_i^t b_j^t$$

Lemma 1 ([2]). *There exists a multi-arm bandit algorithm, A_i, such that for any sequence of observed gains $g_{i,j}^t \in [0,1]$, for any sequence of selected actions k^t, and any action r and parameter $\gamma \in (0,1]$, the expected observed gains is bounded by,*

$$G_{A_i,g^t} \equiv \sum_{t=1}^{T} g_{A_i}^t \equiv \sum_{t=1}^{T} g_{k^t}^t \geq (1-\gamma) \sum_{t=1}^{T} X_{i,r}^t - \frac{N \ln N}{\gamma} - \frac{\gamma}{N} \sum_{t=1}^{T} \sum_{j=1}^{N} X_{i,j}^t \quad (2)$$

We now use Lemma 1. Note, in Auer et al. [2] the action distribution is identical to the selection distribution, i.e. $p^t = q^t$, and the observed and real gain are identical, i.e., $g^t = b^t$. Auer et al. [2] derive the external regret bound by taking the expectation with respect to the action distribution (which is identical to the selection distribution). In our case we will like to separate the real gain from the observed gain.

Let the total observed gain of algorithm A_i be $G_{A_i} = \sum_{t=1}^{T} g_{A_i}^t = \sum_{t=1}^{T} g_{i,k^t}^t$. Since we distribute our gain between the A_i, i.e., $\sum_{i=1}^{N} g_{A_i}^t = b_{Int\_MAB}^t$, we have that $B_{Int\_MAB} = \sum_{t=1}^{T} b_{Int\_MAB}^t = \sum_{i=1}^{N} G_{A_i}$. Since $g_{i,j}^t \in [0,1]$, by Lemma 1, this implies that for any action r we have

$$\begin{aligned}
E_{p^t}[G_{A_i}] &\geq (1-\gamma) \sum_{t=1}^{T} E_{p^t}[X_{i,r}^t] - \frac{N \ln N}{\gamma} - \frac{\gamma}{N} \sum_{t=1}^{T} \sum_{j=1}^{N} E_{p^t}[X_{i,j}^t] \\
&= (1-\gamma) \sum_{t=1}^{T} p_i^t b_r^t - \frac{N \ln N}{\gamma} - \frac{\gamma}{N} \sum_{t=1}^{T} \sum_{j=1}^{N} p_i^t b_j^t \\
&\geq (1-\gamma) B_{i,r} - \frac{N \ln N}{\gamma} - \frac{\gamma}{N} \sum_{j=1}^{N} B_{i,j} \\
&\geq B_{i,r} - O(\sqrt{B_{max} N \ln N} + N \ln N) = B_{i,r} - R^{MAB}(B_{max}, N, T)
\end{aligned}$$

where $B_{i,r} = \sum_{t=1}^{T} p_i^t b_r^t$, $B_{max} = \max_{i,j} B_{i,j}$ and $\gamma = \min\{\sqrt{(N \ln N)/B_{max}}, 1\}$.

Note that the expected benefit of our algorithm is $E[B_{Int\_MAB,b^t}] = \sum_{i=1}^{N} B_{i,i}$. For the regret we like to compare the gain of $B_{i,i}$ to that of $B_{i,r}$, which is the change in our benefit if each time we play action r rather than i. For swap regret, we compare our expected benefit to that of $\sum_{i=}^{N} B_{i,F(i)}$, for some function F. Therefore, we have that for any function $F : \{1, \ldots, N\} \to \{1, \ldots, N\}$,

$$E_{p^t}[B_{Int\_MAB}^T] = \sum_{i=1}^{N} E_{p^t}[G_{A_i}] \geq \sum_{i=1}^{N} B_{i,F(i)} - N \cdot R^{MAB}(B_{max}, T, N).$$

For the case of losses let $b_j^t = 1 - c_j^t$. Then $B_{MAB} = T - L_{MAB}$ and we derive Theorem 4. □

5 External Regret with Time-Selection Functions

We now present a simple online algorithm that achieves a good external regret bound in the presence of time selection functions, generalizing the *sleeping experts* setting. Specifically, our goal is for each action a, and each time-selection function I, that our total loss during the time-steps selected by I is not much more than the loss of a during those time steps (or more generally, the losses weighted by I when $I(t) \in [0,1]$). The idea of the algorithm is as follows. Let $R_{a,I}$ be the regret of our algorithm with respect to action a and time selection function I. That is, $R_{a,I} = \sum_t I(t)(\ell_H^t - \ell_a^t)$. Let $\tilde{R}_{a,I}$ be a less-strict notion of regret in which we multiply our loss by some $\beta < 1$, that is, $\tilde{R}_{a,I} = \sum_t I(t)(\beta\ell_H^t - \ell_a^t)$. What we will do is give to each action a and time selection function I a weight $w_{a,I}$ that is exponential in $\tilde{R}_{a,I}$. We will prove that the sum of our weights never increases, and thereby be able to easily conclude that none of the $\tilde{R}_{a,I}$ can be too large.

Specifically, for each of the N actions and the M time selection functions we maintain a weight $w_{a,I}^t$. We update these weights using the rule $w_{a,I}^{t+1} = w_{a,I}^t \beta^{I(t)(\ell_a^t - \beta\ell_H^t)}$, where ℓ_H^t is the loss of our online algorithm H at time t. (Initially, $w_{a,I}^0 = 1$.) Equivalently, $w_{a,I}^t = \beta^{-\tilde{R}_{a,I}^t}$, where $\tilde{R}_{a,I}^t$ is the "less-strict" regret mentioned above up to time t.

At time t we define $w_a^t = \sum_I I(t) w_{a,I}^t$, $W^t = \sum_a w_a^t$ and $p_a^t = w_a^t / W^t$. Our distribution over actions at time t is p^t.

Claim. At any time t we have $0 \le \sum_{a,I} w_{a,I}^t \le NM$.

Proof. Initially, at time $t = 0$, the claim clearly holds. Observe that at time t we have the following identity,

$$W^t \ell_H^t = W^t \sum_a p_a^t \ell_a^t = \sum_a w_a^t \ell_a^t = \sum_a \sum_I I(t) w_{a,I}^t \ell_a^t. \tag{3}$$

For the inductive step we show that the sum of the weights can only decrease. Note that for any $\beta \in [0,1]$, for $x \in [0,1]$ we have $\beta^x \le 1 - (1-\beta)x$, and for $x \in [-1,0]$ we have $\beta^x \le 1 + (1-\beta)|x|/\beta$.

$$\begin{aligned}
\sum_a \sum_I w_{a,I}^{t+1} &= \sum_a \sum_I w_{a,I}^t \beta^{I(t)(\ell_a^t - \beta\ell_H^t)} \\
&\le \sum_a \sum_I w_{a,I}^t (1 - (1-\beta) I(t) \ell_a^t)(1 + (1-\beta) I(t) \ell_H^t) \\
&\le (\sum_a \sum_I w_{a,I}^t) - (1-\beta)(\sum_{a,I} w_{a,I}^t \ell_a^t) + (1-\beta)(\sum_{a,I} I(t) w_{a,I}^t \ell_H^t) \\
&= (\sum_a \sum_I w_{a,I}^t) - (1-\beta) W^t \ell_H^t + (1-\beta) W^t \ell_H^t \quad \text{(using eqn. (3))}
\end{aligned}$$

$$= (\sum_a \sum_I w^t_{a,I}). \qquad \square$$

Corollary 3. *For every action a and time selection I we have*

$$w^t_{a,I} = \beta^{L_{a,I} - \beta L_{H,I}} \leq MN,$$

where $L_{H,I} = \sum_t I(t)\ell^t_H$ is the loss of the online algorithm with respect to time-selection function I.

A simple algebraic manipulation of the above implies the following theorem

Theorem 5. *For every action a and every time selection function $I \in \mathcal{I}$ we have*

$$L_{H,I} \leq \frac{L_{a,I} + (\log NM)/\log(1/\beta)}{\beta}$$

We can optimize for β in advance, or do it dynamically using [1], establishing:

Corollary 4. *For every action a and every time selection function $I \in \mathcal{I}$ we have*

$$L_{H,I} \leq L_{a,I} + O(\sqrt{L_{min} \log NM} + \log MN),$$

where $L_{min} = \max_I \min_a \{L_{a,I}\}$.

6 Arbitrary time selection and modification rules

In this section we combine the techniques from Sections 3 and 5 to derive a regret bound for the general case where we assume that there is a finite set $\mathcal{I}$ of M time selection functions, and a finite set $\mathcal{F}$ of K modification rules. Our goal is to design an algorithm such that for any time selection function $I \in \mathcal{I}$ and any $F \in \mathcal{F}$, $L_{H,I}$ is not too much larger than $L_{H,I,F}$.

We maintain at time t, a weight $w^t_{j,I,F}$ per action j, time selection I and modification rule F. Initially $w^0_{j,I,F} = 1$. We set

$$w^{t+1}_{j,I,F} = w^t_{j,I,F} \beta^{p^t_j I(t)(\ell^t_{F(j)} - \beta \ell^t_{H,j})},$$

where $W^t_{j,F} = \sum_I I(t) w^t_{j,I,F}$, $W^t_j = \sum_F W^t_{j,F}$, and $\ell^t_{H,j} = \sum_F W^t_{j,F} \ell^t_{F(j)}/W^t_j$.

We use the weights to define a distribution p^t over actions as follows. We select a distribution p^t such that

$$p^t_i = \sum_{j=1}^{N} p^t_j \sum_{F:F(j)=i} W^t_{j,F}/W^t_j. \tag{4}$$

I.e., p is the stationary distribution of the associated Markov chain. Notice that the definition of p implies that the loss of H at time t can either be viewed as $\sum_i p^t_i \ell^t_i$ or as $\sum_j p_j \sum_F (W^t_{j,F}/W^t_j)\ell^t_{F(j)} = \sum_j p^t_j \ell^t_{H,j}$. The following Claim, whose proof is omitted, bounds the magnitude of the weights.

Claim. For every action j, at any time t we have $0 \leq \sum_{I,F} w^t_{j,I,F} \leq MK$

The following theorem (proof omitted) derives the general regret bound.

Theorem 6. *For every time selection $I \in \mathcal{I}$ and modification rule $F \in \mathcal{F}$, we have that*

$$L_{H,I} \leq L_{H,I,F} + O(\sqrt{TN \log MK} + N \log MK)$$

7 Conclusion and open problems

In this paper we give general reductions by which algorithms achieving good external regret can be converted to algorithms with good internal (or swap) regret, and in addition develop algorithms for a generalization of the sleeping experts scenario including both real-valued time-selection functions and a finite set of modification rules.

A key open problem left by this work is whether it is possible to achieve swap-regret that has a logarithmic or even sublinear dependence on N. Specifically, for *external* regret, existing algorithms achieve regret ϵT in time $T = O(\frac{1}{\epsilon^2} \log N)$, but our algorithms for swap-regret achieve regret ϵT only by time $T = O(\frac{1}{\epsilon^2} N \log N)$. We have shown that sublinear dependence is not possible in against an adaptive adversary with swap-regret defined with respect to the actions actually chosen from the algorithm's distribution, but we do not know whether there is a comparable lower bound in the distributional setting (where swap-regret is defined with respect to the distributions p^t themselves), which is the model we used for all the algorithms in this work. In particular, an algorithm with lower dependence on N would imply a more efficient (in terms of number of rounds) procedure for achieving an approximate correlated equilibrium.

References

1. Peter Auer, Nicolò Cesa-Bianchi, , and Claudio Gentile. Adaptive and self-confident on-line learning algorithms. *JCSS*, 64(1):48–75, 2002. A preliminary version has appeared in Proc. 13th Ann. Conf. Computational Learning Theory.
2. Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.
3. R. J. Aumann. Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics*, 1:67–96, 1974.
4. D. Blackwell. An analog ofthe mimimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 6:1–8, 1956.
5. A. Blum. Empirical support for winnow and weighted-majority based algorithms: results on a calendar scheduling domain. *Machine Learning*, 26:5–23, 1997.
6. Nicolò Cesa-Bianchi, Yoav Freund, David P. Helmbold, David Haussler, Robert E. Schapire, and Manfred K. Warmuth. How to use expert advice. In *STOC*, pages 382–391, 1993. Also, *Journal of the Association for Computing Machinery*, 44(3): 427-485 (1997).

7. Nicolò Cesa-Bianchi and Gábor Lugosi. Potential-based algorithms in on-line prediction and game theory. *Machine Learning*, 51(3):239–261, 2003.
8. Nicolò Cesa-Bianchi, Gábor Lugosi, and Gilles Stoltz. Regret minimization under partial monitoring. unpublished manuscript, 2004.
9. Nicolò Cesa-Bianchi, Yishay Mansour, and Gilles Stoltz. Improved second-order bounds for prediction with expert advice. In *COLT*, 2005.
10. W. Cohen and Y. Singer. Learning to query the web. In *AAAI Workshop on Internet-Based Information Systems*, 1996.
11. W. Cohen and Y. Singer. Context-sensitive learning methods for text categorization. *ACM Transactions on Information Systems*, 17(2):141–173, 1999.
12. D. Foster and R. Vohra. Calibrated learning and correlated equilibrium. *Games and Economic Behavior*, 21:40–55, 1997.
13. D. Foster and R. Vohra. Asymptotic calibration. *Biometrika*, 85:379–390, 1998.
14. D. Foster and R. Vohra. Regret in the on-line decision problem. *Games and Economic Behavior*, 29:7–36, 1999.
15. Dean P. Foster and Rakesh V. Vohra. A randomization rule for selecting forecasts. *Operations Research*, 41(4):704–709, July–August 1993.
16. Y. Freund, R. Schapire, Y. Singer, and M. Warmuth. Using and combining predictors that specialize. In *Proceedings of the 29th Annual Symposium on Theory of Computing*, pages 334–343, 1997.
17. Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Euro-COLT*, pages 23–37. Springer-Verlag, 1995. Also, JCSS 55(1): 119-139 (1997).
18. Yoav Freund and Robert E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29:79–103, 1999. A preliminary version appeared in the Proceedings of the Ninth Annual Conference on Computational Learning Theory, pages 325–332, 1996.
19. J. Hannan. Approximation to bayes risk in repeated plays. In M. Dresher, A. Tucker, and P. Wolfe, editors, *Contributions to the Theory of Games*, volume 3, pages 97–139. Princeton University Press, 1957.
20. S. Hart and A. Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68:1127–1150, 2000.
21. S. Hart and A. Mas-Colell. A reinforcement procedure leading to correlated equilibrium. In Wilhelm Neuefeind Gerard Debreu and Walter Trockel, editors, *Economic Essays*, pages 181–200. Springer, 2001.
22. Mark Herbster and Manfred K. Warmuth. Tracking the best expert. In *International Conference on Machine Learning*, pages 286–294, 1995.
23. E. Lehrer. A wide range no-regret theorem. *Games and Economic Behavior*, 42:101–115, 2003.
24. Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, 1994.
25. Gilles Stoltz. Private communication.
26. Gilles Stoltz and Gábor Lugosi. Internal regret in on-line portfolio selection. In *COLT*, 2003. To appear in Machine Learning Journal.
27. Gilles Stoltz and Gábor Lugosi. Learning correlated equilibria in games with compact sets of strategies. submitted to Games and Economic Behavior, 2004.