Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 88 (2007)

Adaptive Algorithms for Online Optimization

Elad Hazan
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120
hazan@us.ibm.com

C. Seshadhri [*]
Princeton University
35 Olden St.
Princeton, NJ 08540
csesha@cs.princeton.edu

Abstract

We study the notion of learning in an oblivious changing environment. Existing online learning algorithms which minimize *regret* are shown to converge to the *average* of all locally optimal solutions. We propose a new performance metric, strengthening the standard metric of regret, to capture convergence to locally optimal solutions. We then describe a series of reductions which transform algorithms for minimizing (standard) regret into adaptive algorithms albeit incurring only poly-logarithmic computational overhead.

We describe applications of this technique to various well studied online problems, such as portfolio management, online routing and the tree update problem. In all cases we explain how previous algorithms perform suboptimally and how the reduction technique gives adaptive algorithms.

Our reductions combine techniques from data streaming algorithms, composition of learning algorithms and a novel use of unbiased gradient estimates.

[*]This work was done while the author was a research intern at the IBM Almaden Research Center

ISSN 1433-8092

1 Introduction

In online optimization the decision maker iteratively chooses a decision without knowledge of the future, and pays a cost based on her decision and the observed outcome. The game theory and machine learning literature has produced a host of algorithms which perform nearly as well as the best single decision in hindsight. Formally, the average *r*egret of the online player, which is the average difference between her cost and the cost of the best strategy in hindsight, approaches zero as the number of game iterations grows. Examples of online optimization scenarios for which such online algorithms were successfully applied include portfolio management [Cov91], online routing [TW03, KV05], and boosting [FS97].

Regret minimization algorithms are particularly useful in scenarios in which the environment variables are sampled from some (unknown) distribution. In such cases, regret minimization algorithms effectively "learn" the environment and approach the optimal strategy. However, if the underlying distribution changes, no such claim can be made.

For example, in the online shortest paths problem (OSP), and online player repeatedly chooses a path in a given graph without knowing the edge congestion in advance. Her cost is the total congestion along the path she chose. Consider the case in which for the first half of the iterations there is a clear shortest path, and for the last half of the iterations there is a different shortest path with the same length. It can be shown that existing efficient algorithms will roughly choose the first path for the first half of the iterations, and move toward an almost uniform distribution over the two optimal paths for the remaining iterations. This behavior, although consistent with regret minimization, attains linear regret compared to the local optimum of the second half of the game iterations, and is clearly suboptimal. We later explain this argument more formally (see Appendix F).

In this paper we address this question of adapting to a changing environment. We argue that the correct measure of performance is *adaptive regret* or the maximum regret on any interval of history. Our main result is an efficiency preserving reduction which transforms *any* low regret algorithm into a low adaptive regret algorithm. We give several variants of online optimization algorithms whose cost on *any* contiguous sequence of iterations is within a small additive term with respect to the *local optimum* - i.e. the cost of the optimal decision on this particular sequence of iterations. In contrast, low regret guarantees small difference in cost only over the entire sequence of iterations and compared to the global optimum.

Our adaptive regret bounds match the lower bounds for regular regret up to logarithmic factors. In addition, the most efficient version of our algorithms have only a logarithmic running time overhead over the most efficient known algorithms and do not impose *any* overhead in terms of "expensive" computations (i.e. shortest path computations for flow problems).

1.1 Our Results

In online optimization, in each round $t = 1, 2, ...$, the decision maker plays a point x_t from a convex domain $K \subseteq \mathbb{R}^n$. A loss function f_t is presented, and the decision maker incurs a loss of $f_t(x_t)$. The standard performance measure is regret, which is the difference between the loss incurred by the online player using algorithm $\mathcal{A}$ and the best fixed optimum in hindsight:

$$\text{Regret}_T(\mathcal{A}) = \sum_{t=1}^{T} f_t(x_t) - \min_{x^* \in K} \sum_{t=1}^{T} f_t(x^*)$$

We consider an extension of the above quantity to measure the performance of a decision maker in a changing environment:

Definition 1. *The adaptive regret of an online convex optimization algorithm $\mathcal{A}$ is defined as the maximum regret it achieves over any contiguous time interval. Formally*

$$\textit{Adaptive-Regret}_T(\mathcal{A}) \triangleq \sup_{I=[r,s]\subseteq[T]} \left\{ \sum_{t=r}^{s} f_t(x_t) - \min_{x_I^* \in K} \sum_{t=r}^{s} f_t(x_I^*) \right\}$$

A crucial point in the above definition is that the comparison in cost is with respect to a *different* optimum for any interval, i.e. x_I^* can vary arbitrarily with I. Obviously adaptive regret is a strict generalization of regret. Intuitively, an algorithm with adaptive regret of at most R converges to the locally optimal solution in each interval of length $\omega(R)$.

We note that Zinkevich's gradient descent algorithm [Zin03], can be shown to attain excellent adaptive regret bounds for certain scenarios (i.e. linear cost functions). However, the regret bounds it attains are very far from optimal for some of the applications we consider (i.e. portfolio management), in which it attains $O(\sqrt{T})$ adaptive regret vs the optimal $O(\log T)$ adaptive regret. In addition, it is not clear how to implement the gradient descent algorithm efficiently for structured problems such as online routing.

In this paper we overcome both of these shortcomings. The following theorem states two reductions from any low regret algorithm to an adaptive regret algorithm, with different efficiency-regret tradeoffs. Throughout the paper the O-notation hides absolute constants.

Theorem 2. *Suppose the functions $f_1, \cdots, f_T$ are α-exp concave (for some constant α) and there exists an algorithm giving $R(T)$ regret with running time V. The running time of algorithm $FLH1$ is $O(VT)$ and $\textit{Adaptive-Regret}_T(FLH1) \leq R(T) + O(\frac{1}{\alpha}\log T)$. The running time of $FLH2$ is $O(V \log T)$ and $\textit{Adaptive-Regret}_T(FLH2) \leq R(T)\log T + O(\frac{1}{\alpha}\log^2 T)$.*

For general convex loss functions, we get a similar theorem -

Theorem 3. *Suppose the functions $f_1, \cdots, f_T$ are convex and bounded by $f(x) \in [0, M]$ in the convex set and there exists an algorithm giving $R(T)$ regret with running time V. The running time of algorithm $FLH1$ is $O(VT)$ and $\textit{Adaptive-Regret}_T(FLH1) \leq R(T) + O(M\sqrt{T \log T})$. The running time of $FLH2$ is $O(V \log T)$ and $\textit{Adaptive-Regret}_T(FLH2) \leq R(T)\log T + O(M\sqrt{T}\log^2 T)$. Furthermore, the amortized running time can be improved to $V + O(\log T)$.*

For convex functions, we actually prove a slightly stronger statement, where we get tradeoffs between a multiplicative factor over the optimal loss and an additive error. Formal statement of the applications of the above theorems appear in section 2.

1.2 Techniques

Much of the novelty in our results is the combination and introduction of techniques, i.e. data streaming ideas, which seem new to the design of online optimization algorithms.

We start by reducing the continuous optimization problem at hand to the discrete realm of experts, which are themselves online optimization algorithms. Although many researchers noted previously that experts algorithms can be composed, we give a very compelling reason to do so - adaptive regret.

To choose amongst the experts, we apply a twist on the well studied Multiplicative Weights method. The set of experts which are the ground set for the MW method, changes online as more experts are added. In order to improve efficiency, we prune the set of experts which are added online. We formally define the properties required of the ground set of experts, and the resulting recipe turns out to be an easily expressible abstract streaming problem. Incorporating the data streaming ideas yields an efficient algorithm.

Next, we further improve the efficiency by introducing unbiased gradient estimates, a technique which was used previously to cope with lack of information in "multi-armed bandit" problems. However, we use such randomization techniques for a completely different purpose - speeding up the algorithm. Usually partial information algorithms incur a large penalty in attainable regret, proportional to the dimension of the problem. However, the combination of the expert-sparsification via streaming and unbiased gradient estimates turns out to incur a negligible penalty in adaptive regret.

1.3 Relation to previous work

The most relevant previous work are the papers of [HW98] and [BW03] on "tracking the best expert". The focus of their work was on the discrete expert setting and exp-concave loss functions. In this scenario, they proved regret bounds versus the best *k-shifting* expert, where the optimum in hindsight is allowed to change its value k times. Freund et al. [FSSW97] generalize this to a setting in which only a subset of the experts make predictions in different iterations.

Our setting differs from this expert setting in several respects. First, we generally consider continuous decision sets rather than discrete. Although it is possible to discretize continuous sets (i.e. the simplex for portfolio management) and apply previous algorithms, such reductions are inefficient. Presumably it might be possible to apply random walk techniques such as Kalai and Vempala [KV03], but that too would be far less efficient than the techniques presented hereby. As for performance guarantees, it is easy to see that our notion of adaptive regret generalizes (and is not equivalent to) regret versus the best *k-shifting* optimum.

Recently there has been some independent work in the information theory community [KS07b, KS07a], which attain related bounds of regret vs. a k-shifting strategy for the portfolio management problem. Our setting is more general, allows for more efficient algorithms and the techniques used are completely different.

2 Applications

For a more detailed exposition, refer to Section A.

Portfolio management. In the universal portfolio management setting, an online investor iteratively distributes her wealth on a set of N assets. After committing to this distribution p_t, the market outcome is observed in a form of a *price relatives vector* r_t and the investor attains utility of $\log(p_t \cdot r_t)$. In his remarkable original paper, Cover [Cov91] analyzes an algorithm called UNIVERSAL which attains

$$\text{Regret}_T(\textsc{Universal}) = \max_{p^*} \sum_{t=1}^{T} \log(p^* \cdot r_t) - \sum_{t=1}^{T} \log(p_t \cdot r_t) = O(n \log T)$$

However, whereas Adaptive-Regret$_T$(UNIVERSAL) $= \Omega(T)$, using Theorem 2, we can prove -

Theorem 4. *There exists an online algorithm $\mathcal{A}$ that for any sequence of price relative vectors $r_1, ..., r_T$, produces wealth distributions $p_1, ..., p_T$ such that*

$$\text{Adaptive-Regret}_T(\mathcal{A}) = O(n \log T)$$

Further, the running time of this algorithm is polynomial in n and T. The running time can be made polynomial in n and $\log T$ with the guarantee Adaptive-Regret$_T(\mathcal{A}) = O(n \log^2 T)$

This bound matches the optimal bound on regular regret, and essentially gives the first theoretical improvement over Cover's algorithm [1].[2]

Online routing and shortest paths. In the online shortest paths (OSP) problem ([TW03, KV05]), an online decision maker iteratively chooses a path in a weighted directed graph without knowing the weights in advance and pays a cost which is the length of her path. Let the graph have m edges, n vertices and weights in the range $[0, 1]$. Here regret is computed with respect to best path in hindsight.

Based on the stronger version of Theorem 3 and using the algorithm from [KV05], we obtain -

Theorem 5. *For the OSP problem, there exists an algorithm $\mathcal{A}$ making* a single *shortest-path computation per round that guarantees -*

$$\text{Adaptive-Regret}_T = O\left(\sqrt{Tmn \log^4 T \log n}\right)$$

This algorithm attains almost optimal adaptive regret. Using FTL ideas the algorithms are easily and efficiently applied to the variety of graph problems considered in [TW03, KV05].

Between static and dynamic optimality for search trees. The classic tree update problem was posed by Sleator and Tarjan in [ST85]. The online decision maker is given a sequence of access requests for items in the set $\{1, ..., n\}$. Her goal is to maintain a binary search tree and minimize the total lookup time and tree modification operations. This problem was looked at from an online learning perspective in [BCK03, KV05]. Using the FTL algorithm for [KV05] and a lazy version of our algorithm, we can prove -

Theorem 6. *Let $a_1, \cdots, a_T$ be accesses made. There is a randomized algorithm $\mathcal{A}$ for the tree update problem that for any contiguous subsequence of queries $I = \{a_r, a_{r+1}, \cdots, a_s\}$, gives the following guarantee with high probability -*

$$\text{cost}_I(\mathcal{A}) \leq \text{cost}_I(\text{best tree for } I) + O(nT^{3/4}(\log T)^{1/4} + n\sqrt{nT})$$

Although the additive term is worse than that of [KV05], it is still sublinear, and we get a very strong version of static optimality.

[1] We note that the running time of Cover's algorithm is exponential. Kalai and Vempala [KV03] gave a polynomial time implementation, which we use for this result. Building on the Online Newton algorithm [HKKA06], we can obtain not only poly-time, but running time which depends only logarithmically on the number of iterations T, albeit introducing dependency in the gradients of the loss functions.

[2] Our guarantee is not to be confused with Singer's "switching portfolios" [Sin]. In his paper Singer deals with switching between single assets, and not CRPs (which is stronger) as is in our case. Our approach is also more efficient.

3 The first step

In this section, we discuss how to get low adaptive regret algorithms, but we do not optimize their running time.

3.1 Exp-concave loss functions

First we consider α-exp concave loss functions, i.e. functions f_t such that $e^{-\alpha f_t}$ is a convex function. The basic algorithm, which we refer to as *Follow-the-Leading-History* (FLH1), is detailed in the figure below. The basic idea is to use many online algorithms, each attaining good regret for a different segment in history, and choose the best one using expert-tracking algorithms. The experts are themselves algorithms, each starting to predict from a different point in history. The meta-algorithm used to track the best expert is inspired by the Herbster-Warmuth algorithm [HW98]. However, our set of experts continuously changes, as more algorithms are considered. This is further complicated in the next section when some experts are also removed.

Algorithm 1 FLH1

```
1: Let $E^1, ..., E^T$ be online convex optimization algorithms. Initialize $p_1^1 = 0$
2: for $t = 1$ to $T$ do
3:     Let $p_t \in \mathbb{R}^t$ be a distribution over the integers $\{1, ..., t\}$.
4:     Set $\forall j \le t$ , $x_t^{(j)} \leftarrow E^j(f_{t-1})$ (the prediction of the $j$'th algorithm).
       play $x_t = \sum_{j=1}^t p_t^{(j)} x_t^{(j)}$.
5:     After receiving $f_t$, set $\hat{p}_{t+1}^{(t+1)} = 0$ and perform update for $1 \le i \le t$ -
           $$\hat{p}_{t+1}^{(i)} = \frac{p_t^{(i)} e^{-\alpha f_t(x_t^{(i)})}}{\sum_{j=1}^t p_t^{(j)} e^{-\alpha f_t(x_t^{(j)})}}$$
6:     Addition step - Set $p_{t+1}^{(t+1)}$ to $1/(t+1)$ and for $i \neq t+1$: $p_{t+1}^{(i)} = (1 - (t+1)^{-1})\hat{p}_{t+1}^{(i)}$
7: end for
```

We prove the following lemma, which shows the strong adaptive regret guarantees of FLH -

Lemma 7. *Suppose that algorithms $\{E^r\}$ attain regret of $R(K)$ over any time interval of length K starting from time r, and have running time V. Then the running time of algorithm FLH1 is $O(VT)$ and Adaptive-Regret$_T$(FLH1) $= R(T) + O(\frac{1}{\alpha}\log T)$.*

Note that each E^r can be implemented using some algorithm that gives (for this case) logarithmic regret, such as the Online Newton Algorithm or standard Follow-The-Leader [HKKA06]. The expert E^r runs the algorithm *starting* from time r - this expert is not considered before time r and definitely satisfies the condition of Lemma 7. The lemma immediately follows from the following stronger performance guarantee:

Lemma 8. *For any interval $I = [r, s]$ in time, the algorithm FLH1 gives $O(\alpha^{-1}(\ln r + \ln |I|))$ regret with respect to the best optimum in hindsight for I.*

By assumption expert E^r gives $R(|I|)$ regret in the interval I (henceforth, the time interval I will always be $[r, s]$). We will show that FLH1 will be competitive with expert E^r in I. To prove Lemma 8, it suffices to prove the following claim.

Claim 9. *For any $I = [r, s]$, the regret incurred by FLH1 in I with respect to expert E^r is at most $\frac{2}{\alpha}(\ln r + \ln |I|)$.*

We first prove the following claim, which gives bounds on the regret in any round and then sum these bounds over all rounds.

Claim 10. *1. For $i < t$, $f_t(x_t) - f_t(x_t^{(i)}) \leq \alpha^{-1}(\ln \hat{p}_{t+1}^{(i)} - \ln \hat{p}_t^{(i)} + 2/t)$*

2. $f_t(x_t) - f_t(x_t^{(t)}) \leq \alpha^{-1}(\ln \hat{p}_{t+1}^{(t)} + \ln t)$

Proof. Using the α-exp concavity of f_t -

$$e^{-\alpha f_t(x_t)} = e^{-\alpha f_t(\sum_{j=1}^t p_t^{(j)} x_t^{(j)})} \quad \geq \quad \sum_{j=1}^t p_t^{(j)} e^{-\alpha f_t(x_t^{(j)})}$$

Taking logarithm,

$$f_t(x_t) \leq -\alpha^{-1} \ln \sum_{j=1}^t p_t^{(j)} e^{-\alpha f_t(x_t^{(j)})}$$

Hence,

$$\begin{aligned} f_t(x_t) - f_t(x_t^{(i)}) \quad &\leq \alpha^{-1}(\ln e^{-\alpha f_t(x_t^{(i)})} - \ln \textstyle\sum_{j=1}^t p_t^{(j)} e^{-\alpha f_t(x_t^{(j)})}) \\ &= \alpha^{-1} \ln \frac{e^{-\alpha f_t(x_t^{(i)})}}{\sum_{j=1}^t p_t^{(j)} e^{-\alpha f_t(x_t^{(j)})}} \\ &= \alpha^{-1} \ln\Big(\frac{1}{p_t^{(i)}} \cdot \frac{p_t^{(i)} e^{-\alpha f_t(x_t^{(i)})}}{\sum_{j=1}^t p_t^{(j)} e^{-\alpha f_t(x_t^{(j)})}}\Big) = \alpha^{-1} \ln \frac{\hat{p}_{t+1}^{(i)}}{p_t^{(i)}} \end{aligned} \qquad (1)$$

The lemma is now obtained using the bounds of Claim 11 below.

□

Claim 11. *1. For $i < t$, $\ln p_t^{(i)} \geq \ln \hat{p}_t^{(i)} - 2/t$*

2. $\ln p_t^{(t)} \geq -\ln t$

Proof. By definition, for $i < t$, $p_t^{(i)} = (1 - 1/t)\hat{p}_t^{(i)}$. Also, $p_t^{(t)} = 1/t$. Taking the natural log of both these inequalities completes the proof. □

We are now ready to prove Claim 9. It will just involve summing up the regret incurred in each round, using Claim 10.

Proof. (Lemma 9) We are looking at regret in I with respect to an expert E^r.

$$\begin{aligned} \sum_{t=r}^{s}(f_t(x_t) - f_t(x_t^{(r)})) \quad &= \quad (f_r(x_r) - f_r(x_r^{(r)})) + \sum_{t=r+1}^{s} (f_t(x_t) - f_t(x_t^{(r)})) \\ &\leq \quad \alpha^{-1}\Big(\ln \hat{p}_{r+1}^{(r)} + \ln r + \sum_{t=r+1}^{s} (\ln \hat{p}_{t+1}^{(r)} - \ln \hat{p}_t^{(r)} + 2/t)\Big) \\ &= \quad \alpha^{-1}(\ln r + \ln \hat{p}_{s+1}^{(r)} + \sum_{t=r+1}^{s} 2/t) \end{aligned}$$

Since $\hat{p}_{s+1}^{(r)} \leq 1$, $\ln \hat{p}_{s+1}^{(r)} \leq 0$. This implies that the regret is bounded by $2\alpha^{-1}(\ln r + \ln |I|)$. □

3.2 General convex loss functions

Here, we explore the case of general convex loss functions. For the sake of simplicity assume that the loss functions f_t are bounded in the domain by $f_t(x) \in [0, M]$. Note that we cannot expect to obtain logarithmic adaptive regret, as even standard regret is known to be lower bounded by $\Omega(\sqrt{T})$. Instead, we derive relative loss bounds, or competitive ratio bounds, and as a special case obtain $O(\sqrt{T} \log T)$ adaptive regret bounds.

We employ the same algorithm as before, and choose our experts accordingly. A small note: instead of taking a convex combination of the experts' predictions, we can also choose experts according to the probability vector p_t. Expert E^i is chosen (in round t) with probability $p_t^{(i)}$. In the latter case, we can prove all the following results with high probability.

Our experts will be any algorithm $\mathcal{A}$ that attains low regret (i.e. the Multiplicative Weights algorithm, Perturbed Follow the Leader). The FLH version for convex functions is almost the same, with a slight change to the multiplicative update rule - the scaling constant changes each iteration. That is, at iteration t, update for $1 \leq i \leq t$ - $\hat{p}_{t+1}^{(i)} = \frac{p_t^{(i)} e^{-\eta_t f_t(x_t^{(i)})}}{\sum_{j=1}^{t} p_t^{(j)} e^{-\eta_t f_t(x_t^{(j)})}}$. The constant η_t will be set according to the bounds we want for strong competitiveness. The main lemma of this section is -

Lemma 12. *If each expert is implemented by a learning algorithm guaranteeing $R(T)$ regret (for T rounds), then the FLH1 algorithm has Adaptive-Regret$_T(FLH1) \leq R(T) + O(M\sqrt{T} \log T)$.*

This lemma immediately follows for the next lemma (proven in appendix).

Lemma 13. *Let $0 < \alpha < \frac{1}{4}$. For any interval $I = [r, s]$, if expert E^r incurs loss L on I, then by setting $\eta_t = -M^{-1} \log(1 - \alpha)$, the loss incurred by FLH1 on I is at most $(1 + \alpha)L + M\alpha^{-1} \ln s$.*

4 Pruning via streaming methods

Our aim now is to implement the above algorithms efficiently and using little space. At time t, the present implementation of FLH1 stores all the experts $E^1, \cdots, E^t$ and has to compute weights for all of them. Let V denote an upper bound on the running time of each expert (for one round). Then the time taken is at least $O(Vt)$. We give an implementation of our algorithm that cuts down this running time, using techniques from streaming algorithms.

The implementation will not depend on whether we deal with exp-concave or general convex functions.

Lemma 14. *Consider the standard implementation of FLH1 and suppose it provides $R(T)$ regret for T rounds. Then FLH2 that has expected $O(V \log T)$ running time (for every round) and has an expected regret of $O(R(T) \log T)$.*

Lemma 15. *Suppose there exists algorithm $\mathcal{A}$ with running time of V per round such that loss$(\mathcal{A}) \leq (1 + \varepsilon)OPT + \varepsilon^{-1}c(T)$ for any sufficiently small ε. Consider the implementation of FLH1 (using $\mathcal{A}$ as experts) and let loss$_I(FLH1)$ be the loss of FLH1 (using $\mathcal{A}$ as experts) on time interval $I = [r, s]$. Suppose loss$_I(FLH1) \leq (1 + \varepsilon)$loss$_I(E^r) + \varepsilon^{-1}d(T)$. The loss of algorithm FLH2 in I is bounded by - loss$_I(FLH2) \leq (1+\varepsilon)OPT(I) + O\left(\frac{\log T}{\varepsilon}(c(T) + d(T))\right)$. The running time of FLH2 is $O(V \log T)$.*

We show that it suffices to store only $O(\log t)$ experts at time t. At time t, there is a working set S_t of experts. In the old implementation of FLH, this set can be thought of to contain $E^1, \cdots, E^t$. For the next round, a new expert E^{t+1} was added to get S_{t+1}. To decrease the sizes of these sets, the efficient implementation will also *remove* some experts. Once an expert is removed, it is never used again (it cannot be added again to the working set). The algorithm will perform the multiplicative update and mixing step only on the working set of experts.

The algorithm FLH2 works exactly the same as standard FLH, with the added *pruning* step. This is the step where certain experts are removed to update the new working set S_{t+1} for round $t+1$.

The problem of maintaining the set of active experts can be thought of, and indeed was considered, as the following abstract data streaming problem. Suppose the integers $1, 2, \cdots$ are being "processed" in a streaming fashion. At time t, we have "read" the positive integers upto t and maintain a very small subset of them S_t. A time t we create S_{t+1} from S_t: we are allowed to add to S_t only the integer $t+1$, and remove any integer already in S_t. Our aim is to maintain a short "sketch" of the data seen so far, i.e. a set S_t which satisfies:

Property 16. 1. *For every positive $s \leq t$, $[s, (s+t)/2] \cap S_t \neq \phi$.*

2. *For all t, $|S_t| = O(\log T)$.*

3. *For all t, $S_{t+1} \backslash S_t = \{t+1\}$.*

There is a randomized construction for these sets given by [GJKK]. Woodruff [Woo07] gave an elegant deterministic construction where the size of $S_t = O(\log t)$. We explain this in the appendix, and for the sake of clarity, do not give details in the algorithm description.

Algorithm 2 FLH2

```
1: At round $t$, there is a set $S_t$ of experts. Abusing notation, $S_t$ will also denote the set of
   indices of the experts. At $t = 1$, $S_t = \{1\}$.
2: for $t = 1$ to $T$ do
3:     Play $x_t = \sum_{j \in S_t} p_t^{(j)} x_t^{(j)}$ (or choose expert $E^j$ with probability $p_t^{(j)}$).
4:     Perform multiplicative update and addition to get vector $\overline{p}_{t+1}$.
5:     Pruning step - Update $S_t$ by removing some experts and adding $t+1$ to get $S_{t+1}$.
6:     For all $i \in S_{t+1}$ -
           $$p_{t+1}^{(i)} = \frac{\overline{p}_{t+1}^{(i)}}{\sum_{i \in S_{t+1}} \overline{p}_{t+1}^{(i)}}$$
7: end for
```

To compute the regret incurred in a given interval, we can only compete with an expert that is present in working set throughout the interval. In such a situation, we get the same regret bounds as before. Our first step is to reprove Claim 11 in the new setting. We restate the claim for convenience.

Claim 17. *For any $i \in S_t$, the following are true - (1) For $i < t$, $\ln p_t^{(i)} \geq \ln \hat{p}_t^{(i)} - 2/t$ (2) $\ln p_t^{(t)} \geq -2 \ln t$*

Proof. The claim is certainly true for $\overline{p}_t^{(i)}$. We note that $p_t^{(i)} \geq \overline{p}_t^{(i)}$ since $\sum_{i \in S_t} \overline{p}_t^{(i)} \leq 1$. □

The proof for the following lemma would be exactly the same of the proofs for Lemmas 9 and 13.

Lemma 18. *Consider some time interval $I = [r, s]$. Suppose that E^r was in the working set S_t, for all $t \in I$. Then the regret incurred in I is at most $R(T)$.*

Given the properties of S_t, we can show that in any interval the regret incurred is small.

Lemma 19. *For interval I, the regret incurred by the FLH2 for any interval I is at most $(R(T) + 1)(\log_2 |I| + 1)$.*

Proof. Let $|I| \in [2^k, 2^{k+1})$. We will prove by induction on k.

base case: For $k = 1$ the regret is bounded by $f_t(x_t) \leq R(T) \leq (R(T)+1)(\log_2 |I|+1)$.

induction step: By the properties of the S_t's, there is an expert E^i in the pool such that $i \in [r, (r + s)/2]$. This expert E^i entered the pool at time i and stayed throughout $[i, s]$. By Lemma 18, the algorithm incurs regret at most $R(T)$ in $[i, s]$.

The interval $[r, i-1]$ has size in $[2^{k-1}, 2^k)$, and by induction the algorithm has regret of $(R(T) + 1)k$ on $[s, i]$. This gives a total of $(R(T) + 1)(k + 1)$ regret on I. □

The running time of FLH2 is bounded by $|S_t|V$. Since $|S_t| = O(\log t)$, we can bound the running time by $O(V \log T)$. This fact, together with Lemma 19, complete the proof of Lemma 14.

5 Incorporating unbiased gradient estimates

We use techniques from [ACBFS02] for dealing with the multi-armed bandit to further improve the running time of FLH for general convex functions. In the present setting, we have then have the perform $|S_t|$ updates, each of which involves an application of a given online algorithm. We will show that an algorithm that makes only a *single* invocation of an expert and still get good adaptive regret bounds. This algorithm works for the general convex function setting.

The general idea is to consider each expert in the set S_t as an arm in the multi-armed-bandit problem. The speedup is obtained since bandit algorithms observe the loss of only a single arm each iteration. Since the regret of bandit algorithms is much inferior in terms of the number of arms as compared to the full information setting, it is crucial at this point to have a small set S_t.

Formally, suppose each expert is implemented using an FTL-like algorithm - given the functions $f_1, \cdots, f_t$, the expert computes some sort of aggregate (in $O(T)$ time) and then outputs the prediction in V time.

Lemma 20. *The algorithm FLH3 obtains adaptive regret of $O(\sqrt{T} \log T)$ (with high probability) and makes a* single *expert invocation in every round. The total running time can be bounded by $VT + O(T \log T)$.*

As a direct consequence of this lemma, we can get a low adaptive regret algorithm for the OSP problem which perform only a single shortest path computation in each round. The key lemma that needs to be proven is -

Lemma 21. *Consider any interval of time $I = [r, s]$ and suppose that expert E^r is present throughout this interval. Then for any constant $c > 10$, with probability $> 1 - T^{-c}$ -*

$$\sum_{t \in I} f_t(x_t) - f_t(x_t^{(r)}) = O(c\sqrt{T} \log T)$$

Algorithm 3 FLH3

```
1: for $t = 1$ to $T$ do
2:     Let $p_t \in \mathbb{R}^{|S_t|}$ be a distribution on the set $S_t$, and let $j_t$ be a random variable with distribution $p_t$. Define $x_t$ to be $x_t^{j_t}$.
       Define the gradient estimate $\tilde{f}_t$ such that $\mathrm{E}[\tilde{f}_t] = f_t$ as follows:
       $$\forall j \in S_t \ . \ \tilde{f}_t(j) = \begin{cases} \frac{f_t(x_t^{(j_t)})}{p_t^{(j_t)}} & j_t = j \\ 0 & j_t \neq j \end{cases}$$
3:     Perform multiplicative update -
       $$u_{t+1}^{(j)} = \frac{p_t^{(j)} \cdot e^{-\eta \tilde{f}_t(x_t^{(j)})}}{Z_t} \ , \ Z_t = \sum_{j \in S_t} p_t^{(j)} \cdot e^{-\eta \tilde{f}_t(x_t^{(j)})}$$
4:     Perform addition step - Set $\overline{u}_{t+1}^{(t+1)}$ to $1/T$ and for $j \neq t+1$, $\overline{u}_{t+1}^{(j)} = (1 - 1/T) u_{t+1}^{(j)}$
5:     Pruning step - Update $S_t$ by removing some experts and adding $t+1$ to get $S_{t+1}$.
6:     For all $j \in S_{t+1}$ -
       $$p_{t+1}^{(j)} = \frac{\overline{u}_{t+1}^{(j)}}{\sum_{j \in S_{t+1}} \overline{u}_{t+1}^{(j)}}$$
7: end for
```

The proof uses techniques from [ACBFS02] to deal with random unbiased estimators for the gradient, and uses martingale arguments to get high concentration. For details, refer to Section E.

References

[ACBFS02] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire. The non-stochastic multi-armed bandit problem. *SIAM Journal of Computing*, 32, 2002.

[BCK03] A. Blum, S. Chawla, and A. Kalai. Static optimality and dynamic search optimality in lists and trees. *Algorithmica*, 36(3), 2003.

[BW03] Olivier Bousquet and Manfred K. Warmuth. Tracking a small set of experts by mixing past posteriors. *J. Mach. Learn. Res.*, 3:363–396, 2003.

[CBL06] Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games.* Cambridge University Press, 2006.

[Cov91] T. Cover. Universal portfolios. *Math. Finance*, 1:1–19, 1991.

[FS97] Yoav Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

[FSSW97] Yoav Freund, Robert E. Schapire, Yoram Singer, and Manfred K. Warmuth. Using and combining predictors that specialize. In *In Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pages 334–343, 1997.

[GJKK] P. Gopalan, T.S. Jayram, R. Krauthgamer, and R. Kumar. Estimating the sortedness of a data stream. In *SODA 2007*.

[HKKA06] Elad Hazan, Adam Kalai, Satyen Kale, and Amit Agarwal. Logarithmic regret algorithms for online convex optimization. In *COLT*, pages 499–513, 2006.

[HW98] Mark Herbster and Manfred K. Warmuth. Tracking the best expert. *Mach. Learn.*, 32(2):151–178, 1998.

[KS07a] S.S. Kozat and A.C. Singer. Universal constant rebalanced portfolios with switching. In *tech report*, 2007.

[KS07b] S.S. Kozat and A.C. Singer. Universal piecewise constant and least squares prediction. In *tech report*, 2007.

[KV03] Adam Kalai and Santosh Vempala. Efficient algorithms for universal portfolios. *J. Mach. Learn. Res.*, 3:423–440, 2003.

[KV05] Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

[OC98] Erik Ordentlich and Thomas M. Cover. The cost of achieving the best portfolio in hindsight. *Mathematics of Operations Research*, 23(4):960–982, 1998.

[Sin] Yoram Singer. Switching portfolios. pages 488–495.

[ST85] D. Sleator and R. Tarjan. Self-adjusting binary search trees. *J. ACM*, 32, 1985.

[TW03] Eiji Takimoto and Manfred K. Warmuth. Path kernels and multiplicative updates. *J. Mach. Learn. Res.*, 4:773–818, 2003.

[Woo07] David Woodruff. personal communications. 2007.

[Zin03] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference (ICML)*, pages 928–936, 2003.

A Applications

Portfolio management. In the universal portfolio management setting, an online investor iteratively distributes her wealth on a set of N assets. After committing to this distribution p_t, the market outcome is observed in a form of a *price relatives vector* r_t and the investor attains utility of $\log(p_t \cdot r_t)$. This formulation is a special case of the online convex optimization setting with a (negative) logarithmic loss function, which is exp-concave.

In his remarkable original paper, Cover [Cov91] analyzes an algorithm called UNIVERSAL which attains

$$\text{Regret}_T(\text{UNIVERSAL}) = \max_{p^*} \sum_{t=1}^{T} \log(p^* \cdot r_t) - \sum_{t=1}^{T} \log(p_t \cdot r_t) = O(n \log T)$$

This was shown to be tight up to constant factors [OC98]. However, similar to the examples in AppendixF, it is easy to construct examples in which the adaptive regret of Cover's algorithm is Adaptive-Regret$_T$(UNIVERSAL) $= \Omega(T)$. Using Theorem 2, we can prove -

Theorem 22. *There exists an online algorithm $\mathcal{A}$ that for any sequence of price relative vectors $r_1, ..., r_T$, produces wealth distributions $p_1, ..., p_T$ such that*

$$\textit{Adaptive-Regret}_T(\mathcal{A}) = O(n \log T)$$

Further, the running time of this algorithm is polynomial in n and T. The running time can be made polynomial in n and $\log T$ with the guarantee Adaptive-Regret$_T(\mathcal{A}) = O(n \log^2 T)$

This bound matches the optimal bound on regular regret, and essentially gives the first theoretical improvement over Cover's algorithm [3].[4]

Online routing and shortest paths. In the online shortest paths (OSP) problem, an online decision maker iteratively chooses a path in a weighted directed graph without knowing the weights in advance and pays a cost which is the length of her path. Let the graph have m edges, n vertices and weights in the range $[0, 1]$. Takimoto and Warmuth [TW03] gave a Multiplicative Weights algorithm which attains (optimal) regret of $O(\sqrt{T})$. Kalai and Vempala [KV05] showed how a simpler approach can give efficient algorithms for OSP and other structured graph problems. Both approaches yield the following more general guarantee:

$$\text{E[total weight]} \leq (1 + \varepsilon)(\text{best weight in hindsight}) + \frac{O(mn \log n)}{\varepsilon}$$

Here best weight in hindsight refers to the total weight of the best single path. Both approaches suffer from the suboptimal behavior explained before, namely the online router may converge to the shortest path of the aggregated graph, which could be very different from the locally optimal path.

Based on Lemma 13, we can construct an algorithm with the following guarantee for *any* interval $I \subseteq [T]$

$$\text{E[total weight on } I] \leq (1+\varepsilon)(\text{best weight in hindsight on } I) + \frac{O(mn \log n \log T + n \log^2 T)}{\varepsilon}$$

Taking $\varepsilon = \frac{1}{\sqrt{Tmn \log^2 T \log n}}$ we obtain

[3]We note that the running time of Cover's algorithm is exponential. Kalai and Vempala [KV03] gave a polynomial time implementation, which we use for this result. Building on the Online Newton algorithm [HKKA06], we can obtain not only poly-time, but running time which depends only logarithmically on the number of iterations T, albeit introducing dependency in the gradients of the loss functions.

[4]Our guarantee is not to be confused with Singer's "switching portfolios" [Sin]. In his paper Singer deals with switching between single assets, and not CRPs (which is stronger) as is in our case. Our approach is also more efficient.

Theorem 23. *For the OSP problem, there exists an algorithm $\mathcal{A}$ with running time polynomial in $m, n, \log T$ that guarantees -*

$$\textit{Adaptive-Regret}_T = O\left(\sqrt{Tmn \log^2 T \log n}\right)$$

This algorithm attains almost optimal adaptive regret. Using FTL ideas the algorithms are easily and efficiently applied to the variety of graph problems considered in [TW03, KV05].

Between static and dynamic optimality for search trees. The classic tree update problem was posed by Sleator and Tarjan in [ST85]. The online decision maker is given a sequence of access requests for items in the set $\{1, ..., n\}$. Her goal is to maintain a binary search tree and minimize the total lookup time and tree modification operations. This problem was looked at from an online learning perspective in [BCK03, KV05]. An algorithm is *statically optimal* if the total time taken by this algorithm is comparable (upto a constant) to the best tree in hindsight. Splay trees are known to be statically optimal, with a constant factor of $3\log_2 3$. Kalai and Vempala [KV05] gave an efficient statically optimal tree algorithm the strong guarantee of low regret (in particular, the constant factor is basically 1). More specifically, they give a randomized algorithm such that -

$$\text{E[cost of algorithm]} \leq \text{(cost of best tree)} + 2n\sqrt{nT}$$

For this, they use a version of the Follow-The-Leader (FTL) which does not give the stronger guarantee of low adaptive regret. We can use our techniques to give such an algorithm, that is basically statically optimal for *every* large enough contiguous subsequence of accesses (note that splay trees are also statically optimal for every contiguous subsequence but the constant multiplicative factor is $3\log_2 3$). We design a lazy version of our algorithm with the following properties -

Lemma 24. *Suppose that for all $x \in K$ and $t \in [T]$, $f_t(x) \in [0, M]$. Let $R(T)$ be an upper bound on the regret of some learning algorithm over a sequence of T functions. There exists a randomized algorithm $\mathcal{A}$, such that with high probability*[5]*, for any $\varepsilon > 0$ less than some sufficiently small constant -*

1. *Adaptive-Regret$_T(\mathcal{A}) \leq R(T) + O(\frac{M\sqrt{T \log T}}{\varepsilon})$*
2. *Throughout the running time of $\mathcal{A}$, $x_t \neq x_{t-1}$ at most εT times.*

We can formulate the tree update problem in the learning setting by considering a point of the domain as a tree. Since it takes $O(n)$ time to update one tree to another (any tree can be changed to another in $O(n)$ rotations), the total modification time is $O(\varepsilon nT)$. Setting $\varepsilon = ((\log T)/T)^{1/4}$, we get -

Theorem 25. *Let $a_1, \cdots, a_T$ be accesses made. There is a randomized algorithm $\mathcal{A}$ for the tree update problem that for any contiguous subsequence of queries $I = \{a_r, a_{r+1}, \cdots, a_s\}$, gives the following guarantee with high probability -*

$$\textit{cost}_I(\mathcal{A}) \leq \textit{cost}_I\textit{(best tree for I)} + O(nT^{3/4}(\log T)^{1/4} + n\sqrt{nT})$$

Although the additive term is worse than that of [KV05], it is still significantly sublinear, and we get a very strong version of static optimality.

[5]the probability of error is $< T^{-2}$

B Multiplicative Weights Lemmas

B.1 General convex loss functions

We give a proof for Lemma 13. Analogous to Lemma 10, the following gives bounds on the regret incurred in any round. For the sake of clarity in the proofs below, we assume that x_t is chosen as a convex combination of the predictions of the various experts. We can also assume that x_t is chosen according to the probability distribution p_t. We then get high probability results, using Chernoff bounds.

Lemma 26. *Suppose that $\forall x \in K$ the value $f_t(x)$ is always positive and bounded by constant M.*

1. *For $i < t$ -*

$$f_t(x_t) \leq \frac{M(\ln \hat{p}^{(i)}_{t+1} - \ln \hat{p}^{(i)}_t + 2/t) + \eta_t M f_t(x^{(i)}_t)}{1 - e^{-\eta_t M}}$$

2.

$$f_t(x_t) \leq \frac{M(\ln \hat{p}^{(t)}_{t+1} + 2 \ln t) + \eta_t M f_t(x^{(t)}_t)}{1 - e^{-\eta_t M}}$$

Proof. We use relative entropy distances. For two n-dimensional vectors, u, v,

$$\Delta(p, q) = \sum_{i=1}^{n} p^{(i)} \ln \frac{p^{(i)}}{q^{(i)}}$$

Conventionally, $0 \ln 0 = 0$. We want to compare performance with respect to the ith expert, and therefore we set $\vec{u}$ to have 1 at the ith coordinate and zero elsewhere.

$$\begin{aligned}
\Delta(u, p_t) - \Delta(u, \hat{p}_{t+1}) &= -\ln p^{(i)}_t + \ln \hat{p}^{(i)}_{t+1} \\
&= -\ln p^{(i)}_t + \ln \frac{p^{(i)}_t e^{-\eta_t f_t(x^{(i)}_t)}}{\sum_{j=1}^{t} p^{(j)}_t e^{-\eta_t f_t(x^{(j)}_t)}} \\
&= -\eta_t f_t(x^{(i)}_t) - \ln\Big(\sum_{j=1}^{t} p^{(j)}_t e^{-\eta_t f_t(x^{(j)}_t)}\Big)
\end{aligned}$$

For $x < M$, we can use the approximation $e^{-\eta_t x} \leq 1 - (1 - e^{-\eta_t M})(x/M)$. This gives us -

$$\begin{aligned}
-\ln p^{(i)}_t + \ln \hat{p}^{(i)}_{t+1} &\geq -\eta_t f_t(x^{(i)}_t) - \ln\Big(\sum_{j=1}^{t} p^{(j)}_t \big(1 - (1 - e^{-\eta_t M})(f_t(x^{(i)}_t)/M)\big)\Big) \\
&= -\eta_t f_t(x^{(i)}_t) - \ln\Big(\sum_{j=1}^{t} p^{(j)}_t - M^{-1}(1 - e^{-\eta_t M}) \sum_{j=1}^{t} p^{(j)}_t f_t(x^{(i)}_t)\Big)
\end{aligned}$$

Noting that p_t is a probability vector and that by convexity, $f_t(x_t) \leq \sum_{j=1}^{t} p^{(j)}_t f_t(x^{(i)}_t)$ -

$$\begin{aligned}
-\ln p^{(i)}_t + \ln \hat{p}^{(i)}_{t+1} &\geq -\eta_t f_t(x^{(i)}_t) - \ln(1 - M^{-1}(1 - e^{-\eta_t M}) f_t(x_t)) \\
&\geq -\eta_t f_t(x^{(i)}_t) + M^{-1}(1 - e^{-\eta_t M}) f_t(x_t) \\
\implies f_t(x_t) &\leq \frac{M(\ln \hat{p}^{(i)}_{t+1} - \ln p^{(i)}_t) + \eta_t M f_t(x^{(i)}_t)}{1 - e^{-\eta_t M}}
\end{aligned}$$

Application of Claim 11 completes the proof. □

Proof. (Lemma 13) For interval I, we sum up the bounds given in Lemma 26. For $\alpha > 0$ sufficiently small, we set $\eta_t = -M^{-1}\ln(1-\alpha)$.

$$\begin{aligned}
\sum_{t=r}^{s} f_t(x_t) &= f_r(x_r) + \sum_{t=r+1}^{s} f_t(x_t) \\
&\leq \frac{M(\ln \hat{p}_{r+1}^{(r)} + 2\ln r) - \ln(1-\alpha) f_r(x_r^{(r)})}{1 - e^{\ln(1-\alpha)}} + \\
&\quad \frac{\sum_{t=r+1}^{s}[M(\ln \hat{p}_{t+1}^{(r)} - \ln \hat{p}_t^{(r)} + 2/t) - \ln(1-\alpha) f_r(x_t^{(r)})]}{1 - e^{\ln(1-2\varepsilon)}} \\
&\leq \alpha^{-1}[M(\ln \hat{p}_{s+1}^{(r)} + \ln s)] - \alpha^{-1}\ln(1-\alpha)\sum_{t=r}^{s} f_r(x_t^{(r)}) \\
&\leq \alpha^{-1}(\alpha + \alpha^2)\sum_{t=r}^{s} f_r(x_t^{(r)}) + M\alpha^{-1}\ln s \\
&= (1+\alpha)\sum_{t=r}^{s} f_r(x_t^{(r)}) + M\alpha^{-1}\ln s
\end{aligned}$$

□

B.2 Variable learning rate

In the previous section we assumed prior knowledge of the number of game iterations T. We now show how to get $O(\sqrt{T}\ln T)$ adaptive regret without knowing T in advance by changing the learning rate.

Lemma 27. *For interval $I = [r, s]$, FLH1 achieves regret of $O(\sqrt{s}\ln s)$ without knowledge of the total time T.*

Lemma 28. *1. For any $i < t$ - $f_t(x_t) - f_t(x_t^{(i)}) \leq \eta_t^{-1}(\ln \hat{p}_{t+1}^{(i)} - \ln \hat{p}_t^{(i)} + \eta_t^2 M^2 + 2/t)$*

2. $f_t(x_t) - f_t(x_t^{(t)}) \leq \eta_t^{-1}(\ln \hat{p}_{t+1}^{(t)} + \eta_t^2 M^2 + \ln t)$

The constant M is an upper bound on $(f_t(x))$.

Proof.

$$\begin{aligned}
\ln p_t^{(i)} - \ln \hat{p}_{t+1}^{(i)} &= \ln p_t^{(i)} - \ln \frac{p_t^{(i)} e^{-\eta_t f_t(x_t^{(i)})}}{\sum_{j=1}^{t} p_t^{(j)} e^{-\eta_t f_t(x_t^{(j)})}} \\
&= \eta_t f_t(x_t^{(i)}) + \ln(\sum_{j=1}^{t} p_t^{(j)} e^{-\eta_t f_t(x_t^{(j)})})
\end{aligned}$$

$$\begin{aligned}
\sum_{\ell=1}^{t} p_t^{(\ell)} \ln \frac{\hat{p}_{t+1}^{(\ell)}}{p_t^{(\ell)}} &= \sum_{\ell=1}^{t} p_t^{(\ell)} \ln \frac{e^{-\eta_t f_t(x_t^{(\ell)})}}{\sum_{j=1}^{t} p_t^{(j)} e^{-\eta_t f_t(x_t^{(j)})}} \\
&= -\eta_t \sum_{\ell=1}^{t} p_t^{(\ell)} f_t(x_t^{(\ell)}) - \ln(\sum_{j=1}^{t} p_t^{(j)} e^{-\eta_t f_t(x_t^{(j)})})
\end{aligned}$$

Using the convexity of f_t, and putting the above equations together, we get -

$$\begin{aligned} f_t(x_t) - f_t(x_t^{(i)}) &= f_t(\sum_{\ell=1}^{t} p_t^{(\ell)} x_t^{(\ell)}) - f_t(x_t^{(i)}) \\ &\leq \sum_{\ell=1}^{t} p_t^{(\ell)} f_t(x_t^{(\ell)}) - f_t(x_t^{(i)}) \\ &= \eta_t^{-1}\Big(\ln \hat{p}_{t+1}^{(i)} - \ln p_t^{(i)} - \sum_{\ell=1}^{t} p_t^{(\ell)} \ln \frac{\hat{p}_{t+1}^{(\ell)}}{p_t^{(\ell)}}\Big) \end{aligned}$$

$$\begin{aligned} -\sum_{\ell=1}^{t} p_t^{(\ell)} \ln \frac{\hat{p}_{t+1}^{(\ell)}}{p_t^{(\ell)}} &= \eta_t \sum_{\ell=1}^{t} p_t^{(\ell)} f_t(x_t^{(\ell)}) + \ln(\sum_{\ell=1}^{t} p_t^{(\ell)} e^{-\eta_t f_t(x_t^{(\ell)})}) \\ &= \eta_t \sum_{\ell=1}^{t} p_t^{(\ell)} f_t(x_t^{(\ell)}) + \ln\big(1 - (1 - \sum_{\ell=1}^{t} p_t^{(\ell)} e^{-\eta_t f_t(x_t^{(\ell)})})\big) \\ &\leq \eta_t \sum_{\ell=1}^{t} p_t^{(\ell)} f_t(x_t^{(\ell)}) - 1 + \sum_{\ell=1}^{t} p_t^{(\ell)} e^{-\eta_t f_t(x_t^{(\ell)})} \\ &\leq \eta_t \sum_{\ell=1}^{t} p_t^{(\ell)} f_t(x_t^{(\ell)}) - 1 + \sum_{\ell=1}^{t} p_t^{(\ell)} (1 - \eta_t f_t(x_t^{(\ell)}) + (\eta_t f_t(x_t^{(\ell)}))^2) \\ &= \eta_t^2 \sum_{\ell=1}^{t} p_t^{(\ell)} f_t(x_t^{(\ell)})^2 \end{aligned}$$

Putting the above together, we get -

$$f_t(x_t) - f_t(x_t^{(i)}) \leq \eta_t^{-1}\Big(\ln \hat{p}_{t+1}^{(i)} - \ln p_t^{(i)} + \eta_t^2 \sum_{\ell=1}^{t} p_t^{(\ell)} f_t(x_t^{(\ell)})^2\Big)$$

Using Claim 11, we have $\ln p_t^{(i)} \geq \ln \hat{p}_t^{(i)} - 2/t$ and that $\ln p_t^{(t)} \geq -t \ln t$. Substituting each of these above, we complete the proof. $\square$

Proof. (Lemma 27) As before, we sum up the regret bounds given by Lemma 28 and set $\eta_t = 1/\sqrt{t}$.

$$\begin{aligned} \sum_{t=r}^{s} (f_t(x_t) - f_t(x_t^{(r)})) &= (f_p(x_p) - f_p(x_p^{(r)})) + \sum_{t=r+1}^{s} (f_t(x_t) - f_t(x_t^{(r)})) \\ &\leq \eta_p^{-1}(\ln \hat{p}_{r+1}^{(r)} + \eta_p^2 M^2 + 2\ln r) + \sum_{t=r+1}^{s} \eta_t^{-1}(\ln \hat{p}_{t+1}^{(r)} - \ln \hat{p}_t^{(r)} + \eta_t^2 M^2 + 2/t) \\ &= M^2 \eta_p + 2\eta_p^{-1} \ln r + \eta_q^{-1} \ln \hat{p}_{s+1}^{(r)} + \sum_{t=r+1}^{s} \ln \hat{p}_t^{(r)} (\eta_{t-1}^{-1} - \eta_t^{-1}) \\ &\quad + \sum_{t=r+1}^{s} M^2 \eta_t + \sum_{t=r+1}^{s} 2/(t\eta_t) \end{aligned}$$

Setting $\eta_t = 1/\sqrt{t}$ -

$$\sum_{t=r}^{s}(f_t(x_t) - f_t(x_t^{(r)})) \leq M^2/\sqrt{r} + 2\sqrt{r}\ln r - \sum_{t=r+1}^{s} \ln \hat{p}_t^{(r)}(\sqrt{t} - \sqrt{t-1}) + (M^2+2)(\sqrt{s} - \sqrt{r})$$

We now provide a lower bound for $\ln \hat{p}_t^{(r)}$, which will allow us to upper bound the regret. Since, in this case, $r < t$ -

$$\hat{p}_t^{(r)} = \frac{p_{t-1}^{(r)} e^{-\eta_{t-1} f_{t-1}(x_{t-1}^{(r)})}}{\sum_{j=1}^{t-1} p_{t-1}^{(j)} e^{-\eta_{t-1} f_{t-1}(x_t^{(j)})}}$$

Since $\forall t$, $\eta_t \leq 1$ and $f_t = \Theta(1)$, $e^{-\eta_{t-1} f_{t-1}(x_t^{(j)})} \in [c_1, c_2]$ (for some positive constants c_1, c_2). We also have that $p_{t-1}^{(r)} = 1/(t-1)$.

$$\hat{p}_t^{(r)} \geq \frac{c_1}{c_2(t-1)\sum_{j=1}^{t-1} p_{t-1}^{(j)}} = O(t^{-1})$$

Therefore, $-\ln \hat{p}_t^{(r)} \leq O(\ln s)$ and we get -

$$\sum_{t=r}^{s}(f_t(x_t) - f_t(x_t^{(r)})) \leq O(\sqrt{s}\ln s)$$

□

C The streaming problem

We now explain Woodruff's solution for maintaining the set $S_t \subseteq [1, n]$ in a streaming manner.

We specify the *lifetime* of integer i - if $i = r2^k$, where r is odd, then the lifetime of i is the interval $2^{k+2} + 1$. Suppose the lifetime of i is m. Then for any time $t \in [i, i+m]$, integer i is *alive* at t. The set S_t is simply the set of all integers that are alive at time t. Obviously, at time t, the only integer added to S_t is t - this immediately proves Property (3). We now prove the other properties -

Proof. (Property (1)) We need to show that some integer in $[s, (s+t)/2]$ is alive at time t. This is trivially true when $t - s < 2$, since $t-1, t \in S_t$. Let 2^ℓ be the largest power of 2 such that $2^\ell \leq (t-s)/2$. There is some integer $x \in [s, (s+t)/2]$ such that $2^\ell | x$. The lifetime of x is larger than $2^\ell \times 2 + 1 > t - s$, and x is alive at t. □

Proof. (Property (2)) For $0 \leq k \leq \lfloor \log t \rfloor$, let us count the number of integers of the form $r2^k$ (r odd) alive at t. The lifetime of these integers are $2^{k+2} + 1$. The only integers alive lie in the interval $[t - 2^{k+2} - 1, t]$. Since all of these integers of this form are separated by gaps of 2^k, there are at most a constant number of such integers alive at t. Totally, the size of S_t is $O(\log t)$. □

D Lazy version

Below we define a lazy version of FLH, called LFLH. We use the “coin-flipping” technique applied in [CBL06] for the “label-efficient prediction” problem. Essentially, we notice that the martingale arguments apply to any low regret algorithm, and even to low adaptive regret algorithms, rather than the multiplicative weights algorithm which they analyze.

LFLH

```
1. Set $\tau = 1$.
2. In round $t$, flip a random $\varepsilon$-balanced coin and obtain the RV $C_t$.
3. If $C_t = 1$ do
    (a) set $g_\tau \triangleq \frac{1}{\varepsilon} f_t$
    (b) update $\tau \leftarrow \tau + 1$.
    (c) Apply FTLH to the function $g_\tau$ to obtain $x_\tau = x_t \leftarrow FLH(g_\tau)$
   Else if $C_t = 0$, set $x_t \leftarrow x_{t-1}$
```

Lemma 29. *Suppose that for all $x \in K$ and $t \in [T]$ $f_t(x) \in [0, M]$. Let $R(T)$ be an upper bound on the regret of the algorithm used to implement FLH1 over a history of length T. Then with high probability, for any $\varepsilon > 0$*

1.
$$\textit{Adaptive-Regret}_T(LFLH) \leq R(T) + O\left(\frac{M\sqrt{T}\log T}{\varepsilon}\right)$$

2. Throughout the running time of LFLH, $x_t \neq x_{t-1}$ at most εT times.

Lemma 30. *Suppose that for all $x \in K$ and $t \in [T]$ $f_t(x) \in [0, M]$. Let $I = [r, s] \subseteq [T]$ be any time interval, and let $R(T)$ be an upper bound on the regret of the algorithm used to implement FLH1 over a history of length T. Then for any $\varepsilon > 0$ and $c > 10$, with probability at least $1 - \frac{1}{T^c}$ it holds that*

$$\textit{Regret}_I(LFLH) \leq R(T) + O\left(\frac{cM\sqrt{T}\log T}{\varepsilon}\right)$$

Proof. Let $f_1, ..., f_T$ be the stream of online cost functions for LFLH. Recall that for each $t \in T$, C_t denotes the outcome of an independent binary coin flip which is one with probability ε. Let

$$\tilde{f}_t \triangleq \begin{cases} 0 & C_t = 0 \\ \frac{1}{\varepsilon} f_t & C_t = 1 \end{cases}$$

The regret of LFLH in a certain interval $I = [r, s] \subseteq [T]$ is

$$\text{Regret}_I = \sum_{t \in I} f_t(x_t) - f_t(x_I^*)$$

Where $x_I^* \triangleq \arg\min_{x\in K} \sum_{t\in I} f_t(x)$. This quantity is a random variable, since the strategies x_t played by LFLH are determined by random coin flips [6]. In order to bound this regret, we first relate it to another random variable, namely

$$Y_I \triangleq \sum_{t\in I} \tilde{f}_t(x_t) - \tilde{f}_t(x_I^*)$$

Observe, that Y_I is the regret of FLH1 on the interval I for the functions $\tilde{f}_t$. Since the bound on the magnitude of the functions $\tilde{f}_t$ is $\frac{M}{\varepsilon}$, we get by Lemma 12 -

$$Y_I \leq R(T) + O(\frac{M}{\varepsilon}\sqrt{T}\log T) \tag{2}$$

We proceed to prove that

$$\Pr[\text{Regret}_I - Y_I \geq 2\frac{\sqrt{cT\log T}}{\varepsilon}] \leq e^{-c\log T} \tag{3}$$

By equations (2) and (3) the Lemma is obtained.

Define the random variable Z_t as

$$Z_t \triangleq f_t(x_t) - f_t(x_I^*) - \tilde{f}_t(x_t) + \tilde{f}_t(x_I^*)$$

Notice that $\sum_{t\in I} Z_t = \text{Regret}_I - Y_I$ and $|Z_t| \leq \frac{4M}{\varepsilon}$. In addition, the sequence of random variables $Z_r, ..., Z_s$ is a martingale difference sequence [CBL06] with respect to the random coin flip variables $C_r, ..., C_s$ [7] since

$$\mathrm{E}[Z_t | C_r, ..., C_{t-1}] = 0$$

The reason is that given all previous coin flips, the point x_t is uniquely determined by the algorithm. The only random variable is $\tilde{f}_t$ and we know that its expectation is exactly f_t. We now use an extension to Hoeffding-Azuma's inequality (see [CBL06]) which implies:

$$\Pr[\sum_{t\in I} Z_t \geq \delta|I|] \leq e^{-\frac{\delta^2|I|\varepsilon^2}{8M^2}}$$

Applying this to our case, with $\delta = \frac{8M\sqrt{c\log T}}{\varepsilon\sqrt{T}}$ we get

$$\Pr[\sum_{t\in I} Z_t \geq \delta|I|] \leq e^{-c\log T} \tag{4}$$

Hence with probability $\geq 1 - \frac{1}{T^c}$ we have

$$\sum_{t\in I} Z_t = \text{Regret}_I - Y_I \leq \delta|I| \leq \frac{cM\sqrt{T\log T}}{\varepsilon}$$

By equation (2) we get that with probability $\geq 1 - \frac{1}{T^c}$ we have

$$\text{Regret}_I \leq R(T) + O(\frac{M}{\varepsilon}\sqrt{T}\log T) + \frac{cM\sqrt{T\log T}}{\varepsilon}$$

$\square$

Lemma 29 follows easily from this lemma and the application of the union bound.

[6]We can henceforth assume that the previous coin tosses $C_1, ..., C_{r-1}$ are arbitrarily fixed, The following arguments hold for any such fixation of previous tosses.

[7]Recall our assumption that the bit flips $C_1, ..., C_{r-1}$ are fixed arbitrarily

E Applying unbiased gradient estimators

We use techniques from [ACBFS02] for dealing with the multi-armed bandit to further improve the running time of FLH for general convex functions. Let us go back to the main idea - at round t, we have a set of carefully chosen experts (in FLH2, this was called S_t) each of whom suggests some point to be played. We have a probability distribution on these experts and play a point accordingly. Once the payoff function f_t is known, this shows us the loss that each expert would have incurred in round t. We have then have the perform $|S_t|$ updates, each of which could be some kind of FTL computation. We will show that only one such computation is required. In other words, we can perform a *single* invocation of an expert and still get good adaptive regret bounds.

Consider each expert to be a different arm of a multi-armed bandit. Choosing an expert to play is equivalent to choosing some arm of the bandit. The function f_t gives a loss on each arm of the bandit (or, on each each expert in S_t). An algorithm was given in [ACBFS02] to get low regret algorithms versus such an adversary. Note that we are not in the bandit setting ourselves, since we are allowed to see the losses for all the arms (since given f_t, the loss on each expert can be determined). Nonetheless, we can use their techniques to get a significant decrease in the running time.

Formally, suppose each expert is implemented using an FTL-like algorithm - given the functions $f_1, \cdots, f_t$, the expert computes some sort of aggregate (in $O(T)$ time) and then outputs the prediction in V time.

Lemma 31. *The algorithm FLH3 obtains adaptive regret of $O(\sqrt{T}\log^2 T)$ (with high probability) and makes a* single *expert invocation in every round. The total running time can be bounded by $VT + O(T\log T)$.*

Proof. The running time bound is easy to see. For each multiplicative update, the only invocation is done for expert E^{j_t}. Once that is done, the weights are updated in $O(\log T)$ time. The adaptive regret bound can be achieved by using Lemma 32 (given below) and a union bound over intervals. □

As a direct consequence of this lemma, we can get a low adaptive regret algorithm for the OSP problem which perform only a single shortest path computation in each round.

Lemma 32. *Consider any interval of time $I = [r, s]$ and suppose that expert E^r is present throughout this interval. Then for any constant $c > 10$, with probability $> 1 - T^{-c}$ -*

$$\sum_{t\in I} f_t(x_t) - f_t(x_t^{(r)}) = O(c\sqrt{T}\log T)$$

We assume that $|f_t| \leq 1$. All vectors considered are $|S_t|$-dimensional, with coordinate indices in S_t. Before proving the main lemma, we prove the following standard lemmas -

Lemma 33. *Let y be an arbitrary vector, z, p_t probability vectors, and u_{t+1} such that -*

$$u_{t+1}^{(j)} = \frac{p_t^{(j)} \cdot e^{-\eta y^{(j)}}}{Z_t} \ , \ Z_t = \sum_{j\in S_t} p_t^{(j)} \cdot e^{-\eta y^{(j)}}$$

Then -

$$y^\top(p_t - z) = \frac{1}{\eta}[\Delta(z, p_t) - \Delta(z, u_{t+1}) + \Delta(p_t, u_{t+1})]$$

Algorithm 4 FLH3

```
1: for $t = 1$ to $T$ do
2:     Let $p_t \in \mathbb{R}^{|S_t|}$ be a distribution on the set $S_t$, and let $j_t$ be a random variable with distribution $p_t$. Define $x_t$ to be $x_t^{j_t}$.
       Define the gradient estimate $\tilde{f}_t$ such that $\mathrm{E}[\tilde{f}_t] = f_t$ as follows:
       $$\forall j \in S_t \ . \ \tilde{f}_t(j) = \begin{cases} \frac{f_t(x_t^{(j_t)})}{p_t^{(j_t)}} & j_t = j \\ 0 & j_t \neq j \end{cases}$$
3:     Perform multiplicative update -
       $$u_{t+1}^{(j)} = \frac{p_t^{(j)} \cdot e^{-\eta \tilde{f}_t(x_t^{(j)})}}{Z_t} \ , \ Z_t = \sum_{j \in S_t} p_t^{(j)} \cdot e^{-\eta \tilde{f}_t(x_t^{(j)})}$$
4:     Perform addition step - Set $\overline{u}_{t+1}^{(t+1)}$ to $1/T$ and for $j \neq t+1$, $\overline{u}_{t+1}^{(j)} = (1 - 1/T) u_{t+1}^{(j)}$
5:     Pruning step - Update $S_t$ by removing some experts and adding $t+1$ to get $S_{t+1}$.
6:     For all $j \in S_{t+1}$ -
       $$p_{t+1}^{(j)} = \frac{\overline{u}_{t+1}^{(j)}}{\sum_{j \in S_{t+1}} \overline{u}_{t+1}^{(j)}}$$
7: end for
```

Proof. First, for any vectors $u, y, z \in \mathbb{R}_+^{|S_t|}$ we have

$$\begin{aligned} \Delta(z,u) - \Delta(z,v) &= \sum_j z^{(j)} \ln \frac{z^{(j)}}{u^{(j)}} - \sum_j z^{(j)} \ln \frac{z^{(j)}}{p^{(j)}} \\ &= \sum_j z^{(j)} \ln \frac{p^{(j)}}{u^{(j)}} = \sum_j (z^{(j)} - p^{(j)}) \ln \frac{p^{(j)}}{u^{(j)}} + \Delta(v,u) \qquad (5) \end{aligned}$$

In our case

$$\begin{aligned} u_{t+1}^{(j)} &= \frac{p_t^{(j)} \cdot e^{-\eta y^{(j)}}}{Z_t} \\ \Longrightarrow \frac{p_t^{(j)}}{u_{t+1}^{(j)}} &= \frac{Z_t}{e^{-\eta y^{(j)}}} \end{aligned}$$

Hence

$$\begin{aligned} \sum_j (z^{(j)} - p_t^{(j)}) \ln \frac{p_t^{(j)}}{u_{t+1}^{(j)}} &= \sum_j (z^{(j)} - p_t^{(j)}) \ln \frac{Z_t}{e^{-\eta y^{(j)}}} \\ &= \eta y^\top (z - p_t) + \ln(Z_t) \cdot (z - p_t)^\top \mathbf{1} \\ &= \eta y^\top (z - p_t) \end{aligned}$$

By equation (5), we have -

$$y^\top (p_t - z) = \eta^{-1} \sum_j (z^{(j)} - p_t^{(j)}) \ln \frac{p_t^{(j)}}{u_{t+1}^{(j)}} = \frac{1}{\eta} [\Delta(z, p_t) - \Delta(z, u_{t+1}) + \Delta(p_t, u_{t+1})]$$

□

Claim 34.
$$\Delta(p_t, u_{t+1}) \le \eta^2 \sum_j (\tilde{f}_t(x_t^{(j)}))^2 p_t^{(j)} \le \eta^2 \sum_j \tilde{f}_t(x_t^{(j)})$$

Proof. Recall as in the previous lemma that

$$\frac{p_t^{(j)}}{u_{t+1}^{(j)}} = \frac{Z_t}{e^{-\eta y^{(j)}}}$$

Hence,

$$\begin{aligned}
\Delta(p_t, u_{t+1}) &= \sum_j p_t^{(j)} \ln \frac{p_t^{(j)}}{u_{t+1}^{(j)}} = \sum_j p_t^{(j)} \ln \frac{Z_t}{e^{-\eta \tilde{f}_t(x_t^{(j)})}} \\
&= \ln Z_t + \eta \tilde{f}_t^\top p_t \\
&= \ln \sum_j p_t^{(j)} e^{-\eta \tilde{f}_t(x_t^{(j)})} + \eta \tilde{f}_t^\top p_t \\
&\le \ln[\sum_j p_t^{(j)} (1 - \eta \tilde{f}_t(x_t^{(j)}) + \eta^2 \tilde{f}_t(x_t^{(j)})^2)] + \eta \tilde{f}_t^\top p_t \qquad \text{if } \eta \tilde{f}_t(i) << 1 \\
&\le \ln[1 - \eta \tilde{f}_t^\top p_t + \eta^2 \sum_j p_t^{(j)} \tilde{f}_t(x_t^{(j)})^2] + \eta \tilde{f}_t^\top p_t \\
&\le \eta^2 \sum_j p_t^{(j)} \tilde{f}_t(x_t^{(j)})^2 \qquad \text{using } \ln(1+x) \le x
\end{aligned}$$

Finally, we observe that $p_t^{(j)} \tilde{f}_t(x_t^{(j)}) \le f_t(x_t^{(j)}) \le 1$. □

We now prove the main lemma of this section.

Proof. (of Lemma 32) In each round t, there is a random variable C_t, which denotes the coin tosses used to select the index j_t. All of these are independent. For the sake of our bounds, we can assume that $C_1, \cdots, C_{r-1}$ are fixed. The function $\tilde{f}_t$ is only a function of C_t, and the probability vector p_t only depends on $C_r, \cdots, C_{t-1}$ (not on C_t). Consider $\tilde{\mathbf{f}}_\mathbf{t}$ as a $|S_t|$-dimensional vector, with jth coordinate $\tilde{f}_t(x_t^{(j)})$. The conditional expected loss of the algorithm in the tth round is $\mathrm{E}_{C_t}[f_t(x_t)] = \sum_j p_t^{(j)} f_t(x_t^{(j)})$. Note that -

$$\mathop{\mathrm{E}}_{C_t}[\tilde{\mathbf{f}}_\mathbf{t} \cdot p_t] = \mathop{\mathrm{E}}_{C_t}[\sum_j \tilde{f}_t(x_t^{(j)}) p_t^{(j)}] = \sum_j p_t^{(j)} \frac{f_t(x_t^{(j)})}{p_t^{(j)}} p_t^{(j)} = \sum_j p_t^{(j)} f_t(x_t^{(j)})$$

Similarly -

$$\mathop{\mathrm{E}}_{C_t}[\tilde{\mathbf{f}}_\mathbf{t} \cdot e^{(r)}] = \mathop{\mathrm{E}}_{C_t}[\tilde{f}_t(x_t^{(r)})] = p_t^{(r)} \frac{f_t(x_t^{(r)})}{p_t^{(r)}} = f_t(x_t^{(r)})$$

Using tha above and previous claims:

$$\begin{aligned}
\sum_{t \in I} \mathop{\mathrm{E}}_{C_t}[f_t(x_t) - f_t(x_t^{(r)})] &= \sum_{t \in I} \mathop{\mathrm{E}}_{C_t}[\tilde{\mathbf{f}}_\mathbf{t}^\top (p_t - e^{(r)})] \\
&= \sum_{t \in I} \frac{1}{\eta} \mathop{\mathrm{E}}_{C_t}[\{\Delta(e^{(r)}, p_t) - \Delta(e^{(r)}, u_{t+1}) + \Delta(p_t, u_{t+1})\}]
\end{aligned}$$

Note that $\Delta(e^{(r)}, u_{t+1}) = -\ln u^{(r)}_{t+1}$. Since $p^{(r)}_{t+1} \geq \hat{u}^{(r)}_{t+1} = (1-1/T)u^{(r)}_{t+1}$, $\ln p^{(r)}_{t+1} \geq \ln u^{(r)}_{t+1} - 2/t$ (the same argument in Claim 11). This implies that $-\Delta(e^{(r)}, u_{t+1}) \leq -\Delta(e^{(r)}, p_{t+1}) + 2/t$. Putting that with the above -

$$\begin{aligned}
\sum_{t\in I} \mathop{\mathrm{E}}_{C_t}[f_t(x_t) - f_t(x^{(r)}_t)] &\leq \sum_{t\in I} \frac{1}{\eta} \mathop{\mathrm{E}}_{C_t}[\Delta(e^{(r)}, p_t) - \Delta(e^{(r)}, p_{t+1}) + \Delta(p_t, u_{t+1})] + \frac{1}{\eta}\sum_t \frac{2}{t} \\
&\leq \frac{1}{\eta} \mathop{\mathrm{E}}_{C_t}[\Delta(e^{(r)}, p_r) - \Delta(e^{(r)}, p_{s+1})] + \frac{1}{\eta}\sum_{t\in I} \mathop{\mathrm{E}}_{C_t}[\Delta(p_t, u_{t+1})] + \frac{2}{\eta}\log T \\
&\leq \frac{1}{\eta} \mathop{\mathrm{E}}_{C_t}[\Delta(e^{(r)}, p_r) - \Delta(e^{(r)}, p_{s+1})] + \eta \sum_{t\in I}\sum_j \mathop{\mathrm{E}}_{C_t}[\tilde{f}_t(x^{(j)}_t)] + \frac{2}{\eta}\log T \quad \text{Claim 34}
\end{aligned}$$

Note that $\Delta(e^{(r)}, p_r) = -\ln p^{(r)} r \leq \ln T$ and $\Delta(e^{(r)}, p_r) \geq 0$. Furthermore, $\mathrm{E}[\tilde{f}_t(x^{(j)}_t)] = f_t(x^{(j)}_t) \leq 1$. Applying these -

$$\sum_{t\in I} \mathop{\mathrm{E}}_{C_t}[f_t(x_t) - f_t(x^{(r)}_t)] \leq \frac{\log T}{\eta} + \eta T \log T + \frac{2}{\eta}\log T$$

Note that it is crucial here that the number of experts ($|S_t|$) is bounded by $\log T$. This comes about because of the careful way of removing experts.

Setting $\eta = 1/\sqrt{T}$, we get that the $\sum_{t\in I} \mathrm{E}_{C_t}[f_t(x_t) - f_t(x^{(r)}_t)] = O(\sqrt{T}\log T)$. We now apply concentration results on martingales to show that the regret is low with very high probability. Consider the random variable Z_t -

$$Z_t = f_t(x_t) - f_t(x^{(r)}_t) - \mathrm{E}_{C_t}[f_t(x_t) - f_t(x^{(r)}_t)]$$

Conditioning on $C_r, \cdots, C_{t-1}$, the expectation is fixed. Because of the independence of the C_t's, we get that $\mathrm{E}[Z_t | C_r, \cdots, C_{t-1}] = 0$. The sequence of random variables $Z_r, \cdots, Z_s$ is a martingale difference sequence [CBL06] with respect to the random coin flip variables $C_r, ..., C_s$. Noting that $|Z_t| \leq 2$, for $\delta > 0$, the Hoeffding-Azuma bound ([CBL06]) gives us -

$$\Pr[\sum_{t\in I} Z_t \geq \delta|I|] \leq e^{-\frac{\delta^2|I|}{4}}$$

Setting $\delta = \sqrt{\frac{4c\log T}{T}}$, we get that with probability $> 1 - T^{-c}$, $\sum_{t\in I} f_t(x_t) - f_t(x^{(r)}_t) = O(c\sqrt{T}\log T)$ $\square$

F Suboptimal behavior of regret minimization algorithms

Consider the online convex optimization framework, in which the decision maker chooses a point from the subset of the real line $x_t \in [-1, 1]$. The convex loss functions are $f_t(x) = (x-1)^2$ for the first $T/2$ iterations, and $f_t(x) = (x+1)^2$ in the last $T/2$ iterations. For the sake of simplicity, consider the “follow-the-leader” (FTL) algorithm which at each iteration predicts the minimum of the aggregated loss function thus far (this algorithm is known to attain $O(\log T)$ regret for this setting [HKKA06]).

The strategy chose by this algorithm will be $x_t = 1$ in the first $T/2$ iterations, and then slowly shift towards $x_t = 0$ in the last $T/2$ iterations. Despite the fact that the total regret is $O(\log T)$, it is easy to see that the adaptive regret is $\Omega(T)$ in the last $T/2$ iterations (where the optimum is obviously -1).

Although we have considered the FTL algorithm, all known logarithmic regret algorithms (Online Newton Step, Cover's algorithm, Exponential Weighting) will behave similarly. In addition, although we described the simplest setting, the same issue arises in the portfolio management and online shortest paths problems described below.

In contrast, our algorithms which attain $O(\log T)$ adaptive regret, initially predict 1, and at $T/2$ start shifting towards -1, and complete this shift at $T/2 + O(\log T)$, thus behaving locally optimal.

ECCC ISSN 1433-8092
http://eccc.hpi-web.de/