Competitive ratio vs regret minimization:
achieving the best of both worlds

Amit Daniely   AMIT.DANIELY@MAIL.HUJI.AC.IL
The Hebrew University of Jerusalem and Google Tel-Aviv

Yishay Mansour   MANSOUR.YISHAY@GMAIL.COM
Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv Israel
and Google Research, Tel Aviv

Editors: Aurélien Garivier and Satyen Kale

Abstract

We consider online algorithms under both the competitive ratio criteria and the regret minimization one. Our main goal is to build a unified methodology that would be able to guarantee both criteria simultaneously. For a general class of online algorithms, namely any Metrical Task System (MTS), we show that one can simultaneously guarantee the best known competitive ratio and a natural regret bound. For the paging problem we further show an efficient online algorithm (polynomial in the number of pages) with this guarantee.

To this end, we extend an existing regret minimization algorithm (specifically, Kapralov and Panigrahy (2011)) to handle movement cost (the cost of switching between states of the online system). We then show how to use the extended regret minimization algorithm to combine multiple online algorithms. Our end result is an online algorithm that can combine a "base" online algorithm, having a guaranteed competitive ratio, with a range of online algorithms that guarantee a small regret over any interval of time. The combined algorithm guarantees both that the competitive ratio matches that of the base algorithm and a low regret over any time interval.

As a by product, we obtain an expert algorithm with close to optimal regret bound on every time interval, even in the presence of switching costs. This result is of independent interest.

Keywords: Online Learning, Competitive Analysis, Paging, Experts, K-Server

1. Introduction

Online algorithms address decision making under uncertainty. They serve a sequence of requests while having uncertainty regarding future requests. We consider the Metrical Tasks Systems (MTS) framework for analyzing online algorithms. In this framework, the online algorithm first receives a request and then decides how to serve it. In order to serve the request, there are two types of costs. A cost for changing the state of the underlying system, and a cost for serving the request from the new state. The cost of the online algorithm is the sum of these two costs.

Historically, the initial dominant form of analysis for such algorithms was to assume a stochastic arrival process for the requests, and analyze the performance of the online algorithm given it (e.g., Coffman and Lueker (1991)). Given a well defined arrival process, there exists a well defined optimal policy and an associated optimal cost. This methodology is quite sensitive to the modeling assumptions, from the specific arrival process, to the assumption about dependencies (e.g., i.i.d. requests), to the robustness when the assumptions are not perfectly met.

Competitive analysis is aimed at addressing those issues and giving a worst case guarantee. Rather than assuming a stochastic arrival process, competitive analysis allows for any request sequence. The main idea is to compare the performance of the online algorithm on the request sequence to that of an omniscient offline algorithm that observes in advance the entire request sequence. The worse case ratio, over all possible request sequences, between the performance of the online algorithm and the performance of the omniscient offline algorithm is the competitive ratio. (See, e.g., Borodin and El-Yaniv (1998).) While for a few online tasks there are algorithms with good (i.e., small) competitive ratios, the competitive analysis approach is often criticized as being too pessimistic. Indeed, for many online tasks there are no algorithms with good competitive ratio.

Another form of analyzing online algorithms is regret minimization. In this approach, originally developed for analyzing online prediction tasks, one fixes a collection of N benchmark algorithms. Then, the loss of the algorithm at hand is compared to the loss of the best performing algorithm in the benchmark. More specifically, the regret is the difference between the cumulative loss of the online algorithm and that of the best benchmark algorithm, and the goal is to have vanishing average regret. Vanishing regret implies matching the performance of the best benchmark algorithm. (See, e.g., Cesa-Bianchi and Lugosi (2006).)

Our main goal is to develop a methodology that would allow to keep best known competitive ratio guarantees, while augmenting them with additional regret minimization guarantees. Specifically, given an online algorithm we will guarantee that our resulting algorithm would have the same competitive ratio of the original algorithm. In addition, we will guarantee that for any time interval we will have a low regret, when compared to the benchmark of algorithms that do not switch states.

Specifically, for any time interval $I = [t_1, t_2]$ the regret is at most $O(\sqrt{|I| \log(TN)})$, where T is the total number of time steps and N is the number states.

This implies, for instance, that if the benchmark makes s state changes, the regret bound would be bounded by $O(\sqrt{sT \log(TN)})$, and for $s \ll T$ the regret is sublinear.

To derive our technical results we introduce a new regret minimization algorithm for the experts problem. Here, the algorithm has to choose one out of N experts at each round. Following its choice, a loss for each expert is revealed, and the algorithm suffers the loss of the expert it chose. The regret of the algorithm is its cumulative loss, minus the cumulative loss of the best expert.

Our algorithm is inspired by the algorithm of Kapralov and Panigrahy (2011). The main benefit of Kapralov and Panigrahy (2011) is that they have essentially zero (exponentially small constant) regret to one expert while having the usual $O(\sqrt{T \log(N)})$ regret with respect to the other experts. Our main technical contribution is to extend the algorithm to work in a *strongly adaptive* and *switching costs* setting. By strong adaptivity we mean that we can bound the regret of any time interval I as a function of the length of I, i.e., $|I|$, rather than the total number of time steps T. By switching cost we mean that we associate a cost with changing experts between time steps. The ability of handling switching costs is critical in order to extend regret minimization results to the competitive analysis framework. Finally, as in Kapralov and Panigrahy (2011), we have a base expert, such that the regret with respect to it is essentially zero. This base expert will model the online competitive algorithm whose performance we like to match. We also maintain an algorithm for each interval size (rounded to powers of two).

Using our regret minimization algorithm we can show that for any metrical task system (MTS) there is an online algorithm that guarantee the minimum between: (1) the competitive ratio times the offline cost plus an additive constant D, the maximum switching cost, and (2) for any time interval the regret with respect to any service sequence with s state changes is at most $O(D\sqrt{s|I| \log(NT)})$.

In addition, the computation time of the algorithm is polynomial in T and N. We note however that in many applications the number of states N is exponential in the natural parameters of the problem. For instance, in the paging problem, where the cache has k out of n pages, the number of states is $N = \Omega(n^k)$. However, we can show how to run our online algorithm in time polynomial in n and k. For the k-server problem we derive an algorithm which is unfortunately not polynomial in n and k. We leave the existence of an efficient algorithm as an open problem, but do point on some possible obstacles. Namely, we show that an efficient sublinear regret algorithm would imply an improved approximation algorithm for the well studied k-median problem.

Related Work. Adaptivity has attained much attention in the online learning literature over the years (an incomplete list includes Herbster and Warmuth (1998); Bousquet and Warmuth (2003); Blum and Mansour (2007); Hazan and Seshadhri (2007); Kapralov and Panigrahy (2011); Cesa-Bianchi et al. (2012); Adamskiy et al. (2012); Panigrahy and Popat (2013); Luo and Schapire (2015); Daniely et al. (2015); Altschuler and Talwar (2018). Another related, but somewhat orthogonal line of work Zinkevich (2003); Hall and Willett (2013); Rakhlin and Sridharan (2013); Jadbabaie et al. (2015) studies *drifting environments*. Switching costs were studied in Kalai and Vempala (2005) and attained considerable interest recently (e.g, Geulen et al. (2010); Cesa-Bianchi et al. (2013); Dekel et al. (2014)). Connections between regret-minimization and competitiveness has been previously studied. In particular Blum et al. (2002); Blum and Burch (2000); Abernethy et al. (2010); Buchbinder et al. (2012) focus on algorithmic techniques that simultaneously apply to MTS and experts, and also show regret minimization algorithms for MTS an other online problems. Another example is Andrew et al. (2013) that studies tradeoffs between regret minimization and competitiveness for a certain online convex problem.

Technically, the work of Kapralov and Panigrahy (2011) is quite close to our work. They showed an algorithm for the two experts problem that has essentially no regret w.r.t. one of the experts, and nearly optimal regret ($\tilde{O}(\sqrt{T})$) w.r.t. the second expert. They also showed that this result implies an algorithm with strongly adaptive guarantees in the N-experts problem. Namely, they showed an algorithm with nearly optimal regret on every *geometric* time interval. Our main contribution, regarding regret minimization, is to handle switching costs. In addition, we extend the regret bound to hold on all intervals.

2. Model and Results

2.1. Strongly adaptive expert algorithms in the presence of switching costs

We consider an online setting where there are N experts and a switching cost of $D \geq 0$, which we call N-experts D-switching cost problem. There are T time steps, and at time $t = 1, \ldots, T$, the online algorithm chooses an expert $i_t \in [1, N]$. Then, the adversary reveals a loss vector $l_t = (l_t(1), \ldots l_t(N)) \in [0, 1]^N$. The loss of the algorithm at time t is $l_t(i_t)$. In addition, if i_t is different from i_{t-1}, the algorithm suffers an additional switching cost of D. This implies that the total loss of the algorithm is $\sum_{t=1}^{T} l_t(i_t) + D \sum_{t=2}^{T} 1[i_t \neq i_{t-1}]$.

We assume that the adversary is oblivious, i.e., the sequence of losses is chosen before the first time step. Likewise, we assume that the time horizon (denoted by T) is known in advance. Given a time interval $I = \{i_0 + 1, ..., i_0 + k\}$ we say that the online algorithm has a regret bound of $R(I)$

in I if for any sequence of losses we have

$$\mathbb{E}\left[\sum_{t\in I} l_t(i_t) + D \sum_{t\in I\setminus\{i_0+1\}} 1[i_t \neq i_{t-1}]\right] \leq \min_{i\in[1,N]} \sum_{t\in I} l_t(i) + R(I) ,\tag{1}$$

where $1[\cdot]$ is the indicator function and the expectation is over the randomization of the online algorithm. Note that we sum the losses only for $t \in I$ and the switching costs only for $t \in \{i_0 + 2, ..., i_0 + k\}$. When we refer to the *regret of the algorithm in I* we mean the minimal $R(I)$ for which inequality (1) holds. We say that the algorithm has a regret bound of $R(T)$ if it has a regret bound of $R(T)$ in the interval $[1, T]$. Our first result presents an algorithm whose regret on any interval I is close to optimal, even when switching costs are present. Specifically, we show that

Theorem 1 *There is an N-experts D-switching cost algorithm with $O\left(N\log(T)\right)$ per-round computation, whose regret on every interval $I \subseteq [T]$ is at most $O\left(\sqrt{(D+1)|I|\log(NT)}\right)$*

We next show that Theorem 1 is tight, up to the dependency on D. Recall that even when ignoring switching costs and considering only interval I the regret is $\Omega\left(\sqrt{|I|\log(N)}\right)$ (e.g., Cesa-Bianchi et al. (1997)). The following theorem improves the lower bound by showing that in order to have a regret bound for *any* interval there is an additional log factor of T (even for $D = 0$).

Theorem 2 *For every N-experts algorithm there is segment I with regret $\Omega\left(\sqrt{|I|\log(NT)}\right)$.*

2.2. Metrical Task systems and competitive analysis

MTS model: An MTS is a pair $(\mathcal{X}, \mathcal{L})$ where $\mathcal{X}$ is an N-points (pseudo-)metric space and $\mathcal{L} \subset [0,1]^{\mathcal{X}}$ is a collection of possible loss vectors. Each $x \in \mathcal{X}$ represents a state of the MTS, and the distance function $d(x_1, x_2)$ describes the cost of moving between states $x_1, x_2 \in \mathcal{X}$. At each step $t = 1, \ldots, T$ the online algorithm is first given a loss vector $l_t \in \mathcal{L}$ and then chooses a state $i_t \in \mathcal{X}$. Following that, the algorithm suffers a loss of $l_t(i_t) + d(i_t, i_{t-1})$. Therefore, for a sequence of T steps the toal loss would be $\sum_{t=1}^T l_t(i_t) + d(i_t, i_{t-1})$. As in the experts problem, we assume that the adversary is oblivious and that the time horizon (denoted by T) is known in advance.

MTS versus Experts: There are three differences between the MTS model and the experts model: (i) In the MTS problem the algorithm observes the loss before it chooses an action while in the expert problem the algorithm first selects the action and only then observes the losses. (ii) In MTS the switching costs are dictated by an arbitrary metric, while in our experts setting the switching cost is always the same. (iii) In the expert problem the losses can be arbitrary, while in MTS the losses are restricted to be elements of $\mathcal{L}$.

Competitive Analysis: One of the classical measures of online algorithms is *competitive ratio*. An online algorithm has a competitive ratio $\alpha \geq 1$ if, up to an additive constant and for any sequence of losses, the loss of the algorithm is bounded by α times the loss of the best possible sequence of actions. In other worlds, the loss of the online algorithm is competitive with the optimal offline algorithm that is given the sequence of losses (requests) in advance. Formally, there is a constant $\beta > 0$ (independent of T) such that for any sequence of losses $l_1, \ldots, l_T$ we have that

$$\mathbb{E}\left[\sum_{t=1}^T l_t(i_t) + \sum_{t=2}^T d(i_t, i_{t-1})\right] \leq \alpha \min_{i_1^*, \ldots, i_T^* \in \mathcal{X}} \left(\sum_{t=1}^T l_t(i_t^*) + \sum_{t=2}^T d(i_t^*, i_{t-1}^*)\right) + \beta .\tag{2}$$

Classical results by Borodin et al. (1992) provide, for any MTS, a deterministic algorithm with competitive ratio $2N - 1$. This is optimal, in the sense that there are MTSs with no deterministic algorithm with a better competitive ratio. For randomized algorithms, the best known competitive ratio Fiat and Mendel (2003); Bartal et al. (1997) for a general MTS is $O\left(\log^2(N)\log\log(N)\right)$, while the best known lower bound Bartal et al. (2005, 2006) is $\Omega\left(\frac{\log(N)}{\log\log(N)}\right)$.

Regret for MTS: The notions of regret and regret on a time interval for MTS are defined in a fashion similar to the experts problem. Given time interval $I = \{i_0 + 1, ..., i_0 + k\}$ we say that the algorithm has a regret bound of $R(I)$ in I if for any sequence of losses we have

$$\mathbb{E}\left[\sum_{t \in I} l_t(i_t) + \sum_{t \in I \setminus \{i_0 + 1\}} d(i_t, i_{t-1})\right] \leq \min_{i \in [1,N]} \sum_{t \in I} l_t(i) + R(I) \,, \tag{3}$$

where the expectation is over the randomization of the algorithm. Likewise, we say that the algorithm has a regret bound of $R(T)$ if it has a regret bound of $R(T)$ in the interval $[1, T]$.

Competitive ratio versus regret minimization: Both competitive analysis and regret minimization are measuring the quality of an online algorithm compared to an offline counterpart. There are two major differences between the two approaches. The first is the type of benchmark used for the comparison: Competitive analysis allows an arbitrary offline algorithm while regret minimization is limiting the benchmark to the best static expert, i.e., selecting the same expert at every time step. The second difference is the quantitative comparison criteria. While in competitive analysis the comparison is multiplicative, in the regret analysis the comparison is additive.

From Theorem 1 with switching costs of $D = \max_{x,y \in X} d(x, y)$ it is not hard to conclude that:

Corollary 3 *For any MTS there is an algorithm whose regret on any interval $I \subseteq [1, T]$ is $O\left(\sqrt{(D+1)|I|\log(NT)}\right)$.*

Based on a variant of Theorem 1 (namely, Theorem 12 below) we show that,

Corollary 4 *Given an online algorithm $\mathcal{A}_{\text{base}}$ for a MTS $\mathcal{M}$ with competitive ratio $\alpha \geq 1$, there is an online algorithm $\mathcal{A}$ for $\mathcal{M}$ such that*

- *$\mathcal{A}$ has a competitive ratio of α*

- *The regret of $\mathcal{A}$ on every interval $I \subseteq [T]$ is $O\left(D\sqrt{|I|\log(T)} + \sqrt{(D+1)|I|\log(NT)}\right)$.*

Furthermore, the per-round computational overhead of $\mathcal{A}$ on top of $\mathcal{A}_{\text{base}}$ is $O(N\log(T))$.

2.3. Paging and k-sever

The result regarding the MTS framework (Corollary 4) gives a general methodology to achieve the best of both world: guaranteeing a low regret with respect to the best static solution and at the same time guaranteeing a good competitive ratio. One drawback of Corollary 4 is the dependency on the number of states, N. While for an abstract setting, such as MTS, one should expect at least a linear dependency on the number of states N, in many concrete application this number is exponential in the natural parameters of the problem. In this section we discuss two such cases, the paging problem and the k-server problem. For the paging problem we show how to overcome the computation issue. For the k-server problem we show some possible computational limitations.

Paging. In the online paging problem there is a set P of n memory pages, out of which k can be in the cache at a given time. At each time we have a request for a page, and if the page is not located in the cache we have a *cache miss*. In this case, the algorithm has to fetch the page from memory to the cache and incur a unit cost. If when it fetches the page, the cache is full (has k pages) then it also has to evict a page. The algorithm tries to minimize the number cache misses.

It is fairly straightforward to model the paging problem as an MTS. The states of the MTS will be all possible configuration in which the cache is full, i.e., all $C \subset P$ such that $|C| = k$. (We can assume without loss of generality that the cache is always full.) The number of states of the MTS is $N = \binom{n}{k}$. We need to define a metric between the states and possible loss functions. Given two configurations C_1 and C_2 let $d(C_1, C_2) = |C_1 \setminus C_2|$. First, note that since the cache is always full, the distance is symmetric. Second, moving from cache C_1 to cache C_2 involves fetching the pages $C_1 \setminus C_2$ and evicting the pages $C_2 \setminus C_1$, and has cost $|C_1 \setminus C_2|$. Now we need to define the possible loss function. We have a possible loss function ℓ_i for each request page $i \in P$. For a cache C, if $i \in C$ then $\ell_i(C) = 0$ and if $i \notin C$ we have $\ell_i(C) = 2$. Clearly when $i \in C$ we do not have any cost, but when $i \notin C$ we like to allow the online algorithm to "stay" in state C, which would involve fetching page i (unit cost) while evicting some page $j \in C$ and then fetching back page $j \in C$ (another unit cost). This explains why we charge two when $i \notin C$.

The paging problem has been well studied as one of the prototypical online problems. The best possible competitive ratio for deterministic algorithms is k, and is achieved by various algorithms Sleator and Tarjan (1985) including Least Recently Used (LRU) and First In First Out (FIFO). For randomized algorithms, the randomized marking algorithm enjoys a competitive ratio of $2H_k = 2\sum_{i=1}^{k} \frac{1}{k} \approx 2\log(k)$, and is optimal up to a multiplicative factor of 2 Fiat et al. (1991). By Corollary 4 we have

Corollary 5 *There is a paging algorithm such that*

- *Its competitive ratio is $2H_k$*

- *Its regret on any interval I is*

$$O\left(k\sqrt{|I|\log(T)} + \sqrt{k|I|\log\left(\binom{n}{k}T\right)}\right) = O\left(k\sqrt{|I|\log(nT)}\right)$$

Let us rephrase Corollary 5 in terms of the paging problem. It guarantees an online algorithm for which the number of cache misses is at most a $2H_k$ larger than what is achieved by the optimal offline schedule. Likewise, for long enough segments, i.e., longer than $\Omega(k^2\log(nT))$, it guarantees that the number of cache misses is not much larger compared to the best single "fixed" cache C^*. Recall that for a fixed cache C^*, whenever we have a request for a page $i \notin C^*$ we first fetch i evicting $j \in C^*$ and then evict i and fetch back j, for a total cost of two.

We note that there are many natural cases where a fixed cache is either optimal or near optimal. For example, if the page requests are distributed i.i.d. then there is a cache configuration C^* which minimizes the probability of a cache miss. Namely, the configuration that has in the cache the k pages with highest probability. This implies that this fixed cache strategy is optimal, up to a multiplicative factor of 2 in this distributional setting. Note that we do not need a single cache configuration for all T time steps, but rather only for the interval I.

As discussed before, the main drawback of a naive application of Corollary 4, i.e., Corollary 5, is the running time of the algorithm which scales with the number of states $N = O(n^k)$. Our main additional contribution for the paging problem is to make the online algorithm efficient. Namely, we show the following theorem.

Theorem 6 *There is an online paging algorithm with per-round runtime of* $\mathrm{poly}(n, \log(T))$ *that enjoys a regret of* $O(\sqrt{kT \log(n)})$.

Based on the algorithm of Theorem 6 and a variant of Corollary 4 (Theorem 12) we obtain an efficient algorithm with guarantees as in Corollary 5.[1]

Corollary 7 *There is a paging algorithm with per-round runtime of* $\mathrm{poly}(n, \log(T))$ *such that*

- *Its competitive ratio is* $2H_k$

- *Its regret on any interval I is at most* $O\left(k\sqrt{|I| \log(nT)}\right)$

k**-server.** In the online k-server problem there is a set X of n locations and a set of k servers, each located at some location $x \in X$. There is a metric define over X, namely, $d(x_1, x_2)$ is the distance between the locations $x_1, x_2 \in X$. We assume that for all $x_1, x_2 \in X$, $d(x_1, x_2) \leq 1$. At each time t we have a request for a location $x_t \in X$. If there is a server located at x_t we have a zero cost, and otherwise we have to move one of the servers to location x_t. The cost of the online algorithm is the sum of the distances the servers have traversed. We note that paging is a special case of the k-server problem where X is the uniform metric space (i.e., $d(x, y) = 1[x \neq y]$).

Again, it is fairly straightforward to model the k-server problem as an MTS. The number of states of the MTS will be all possible locations of the k servers, i.e., X^k. This implies that the number of states of the MTS is $N = n^k$. Again, note that the number of states is exponential in k.

We now need to define a metric between the states and possible loss functions. First we extend the distance function $d(\cdot, \cdot)$ to configuration in X^k. Given two configurations $C_1, C_2 \in X^k$ let $d(C_1, C_2)$ is the minimum weight matching between the k locations in C_1 and the k locations in C_2, where the weight between two locations $x_1, x_2 \in X$ is their distance $d(x_1, x_2)$. By definition we can move from configuration C_1 to C_2 having cost $d(C_1, C_2)$ by utilizing the minimum weight matching. Now we need to define the possible loss functions. We have a possible loss function ℓ_x for each location $x \in X$. For a configuration $C \in X^k$, if $x \in C$ then $\ell_x(C) = 0$ and if $x \notin C$ we have $\ell_x(C) = 2d(C, x)$, where $d(C, x) = \min_{y \in C} d(y, x)$. Clearly when $x \in C$ we do not have any cost, but when $x \notin C$ we like to allow the online algorithm to "stay" in configuration C, which would involve moving a server to location x (distance $d(C, x)$) and back (another distance $d(C, x)$). This explains why we charge $2d(C, x)$ when $x \notin C$.

The k server problem has been extensively studied in the online algorithms literature. The best known competitive ratio Koutsoupias and Papadimitriou (1995) for a deterministic algorithm is $2k - 1$. As for lower bounds, the best known is the paging lower bound of k, and it is conjectured to be tight. As for randomized algorithms the best known lower bound is again the paging lower bound of H_k. As for upper bounds, a recent result by Bubeck et al. (2017) improved Bansal et al. (2015) and showed an $O(\log^2(k))$-competitive algorithm. By Corollary 4 we get that

---

1. Note that that compared to theorem 6 there is are slight differences in the regret term. The additional factor of $O(\sqrt{k})$ arises from the need to maintain the competitive ratio of $2H_k$. The additional logarithmic factor in T is needed in order to support the adaptivity over all interval.

Corollary 8 *For any metric space X there is a k-server algorithm such that*

- *Its competitive ratio is $\min\left(2k - 1, O(\log^2(k))\right)$*

- *Its regret on any interval I is $O\left(k\sqrt{|I|\log(T)} + \sqrt{k|I|\log(n^k T)}\right) = O\left(k\sqrt{|I|\log(nT)}\right)$*

As in the case of paging, as the number of configurations is n^k, and a naive application of Corollary 4 will result with a rather inefficient algorithm. Unlike paging problem, we are unable to derive an efficient online algorithm with similar guarantees. We are able to show that exhibiting such an online algorithm would have interesting implications. In section 7 we show that an online algorithm which achieves such guarantees and that runs in time polynomial in n will result with a 2-approximation algorithm for the k-median problem.

Theorem 9 *If there is an online algorithm for the k-server problem that has regret $poly(n)T^{1-\mu}$, then for every $\epsilon > 0$ there is an polynomial time algorithm that achieves a $2 + \epsilon$ approximation for the k-median problem.*

Algorithm as in Theorem 9 will improve on the best known 2.675-approximation ratio Byrka et al. (2015) of this well studied problem. It is worth noting that the best known hardness of approximation results Jain et al. (2002) for k-medians only rule out approximation ratio of $1 + \frac{2}{e} - \epsilon \approx 1.7358$.

3. Overview over the Proofs

In this section we sketch our algorithms and the correctness proof, namely proving Theorem 1.

Reducing to linear optimization. Our first step is to consider an equivalent continuous version of the N-expert problem, which previously appeared in the literature in the context of MTS (see for instance Blum and Burch (2000)). It turns out that the N-experts D-switching cost problem is equivalent to the following linear optimization problem. At each step $t = 1, \ldots, T$ the player chooses $x_t \in \Delta^N$. Then, the adversary chooses a loss vector $l_t \in [0,1]^N$ and the player suffers a loss of $\langle l_t, x_t \rangle + D\|x_t - x_{t-1}\|_{TV} = \sum_{i=1}^{N} l_t(i)x_t(i) + \frac{D|x_t(i) - x_{t-1}(i)|}{2}$, where $\| \cdot \|_{TV}$ is the total-variation distance.

The intuition is that at time $t - 1$ the player has an action distributed according to x_{t-1}. At time t the player needs to sample an action from x_t. This can be done in a way that the probability of making a switch is exctly $\|x_t - x_{t-1}\|_{TV}$.

An overview. We first discuss why previous approaches fail to achieve our goal. Previous results for the switching costs setting were proved by using an algorithm whose number of switches, on every input sequence, is upper bounded by the desired regret bound. In this case, the switching costs can be simply absorbed in the regret term. We next show that this strategy cannot work in the strongly adaptive setting. To this end, let us consider the following scenario. There are two experts and $D = 1$. In the first $T^{\frac{1}{10}}$ steps the first expert has 0 loss, while the second has a loss of 1 at each step. Then, this is flipped every $T^{\frac{1}{10}}$ rounds. Namely, the loss sequence is

$$\underbrace{(0,1), (0,1), \ldots, (0,1)}_{T^{\frac{1}{10}} \text{ times}}, \ \underbrace{(0,1), (0,1), \ldots, (0,1)}_{T^{\frac{1}{10}} \text{ times}}, \ldots, \underbrace{(1,0), (1,0), \ldots, (1,0)}_{T^{\frac{1}{10}} \text{ times}},$$

Suppose now that $\mathcal{A}$ is a strongly adaptive algorithm operating on this sequence. Concretely, let us assume that $\mathcal{A}$ is guaranteed to have a regret of $\sqrt{|I|\log(T)}$ on every time interval $I \subset [T]$. Let us

first consider the operation of the algorithm in blocks of the form $I_i = [iT^{\frac{1}{10}} + 1, (i+1)T^{\frac{1}{10}}]$. In each block I_{2j}, $\mathcal{A}$ will have to give the first expert a weight $\geq \frac{3}{4}$ at least once. Indeed, otherwise, its regret would be $\geq \frac{1}{4}T^{\frac{1}{10}} \gg \sqrt{T^{\frac{1}{10}}\log(T)}$. Similarly, in each block I_{2j+1}, $\mathcal{A}$ will have to give the second expert a weight $\geq \frac{3}{4}$ at least once. Let us now consider the number of switches during the the entire run. By the arguments above, the number of switches $\mathcal{A}$ makes will be at least $\frac{1}{2}T^{\frac{9}{10}} \gg \sqrt{T\log(T)}$.

In this paper we take a different approach, based on the algorithm of Kapralov and Panigrahy (2011) for the two experts problem without switching costs, which achieves essentially zero regret w.r.t. to the first expert, while still maintaining optimal asymptotic regret w.r.t. the second, i.e., $O(\sqrt{T})$. Concretely, given a parameter $0 < Z < \frac{1}{e}$, the regret w.r.t. the first expert is ZT, while the regret w.r.t. the second is $\sqrt{64T\log\left(\frac{1}{Z}\right)} + ZT + 4$. For $Z = \sqrt{(\log T)/T}$ we get the usual regret $O(\sqrt{T\log T})$, but we will aim for $Z = 1/T$ which will have a regret of 1 to the first expert and a regret of $O(\sqrt{T\log T})$ to the second expert.

We extend the analysis of Kapralov and Panigrahy (2011) and show that the algorithm has similar regret bound even in the presence of switching costs. Furthermore, we show that a variant of this algorithm (obtained by adding a certain projection) enjoys such a regret bound on any time interval. Concretely, we prove that

Theorem 10 *There is an algorithm for the 2-experts D-switching cost problem, that given parameters $Z \leq \frac{1}{e}$ and $\tau \geq 1$ has the following regret bounds*

- *For any time interval $I \subseteq [T]$, the regret of the algorithm w.r.t. expert 0 is at most*
  $$\min\left\{\sqrt{D}TZ, \sqrt{16D\tau\log\left(\frac{1}{Z}\right)} + 2\sqrt{D} + \sqrt{D}|I|Z\right\}$$

- *For every time interval $I \subseteq [T]$ of length $\leq \tau$, the regret of the algorithm w.r.t. expert 1 is at most $\sqrt{64D\tau\log\left(\frac{1}{Z}\right)} + 4\sqrt{D} + \sqrt{D}\tau Z$*

To better understand Theorem 10, consider setting $Z = 1/(\sqrt{D}T)$ and $\tau = T$:

Corollary 11 *There is an algorithm for the 2-experts D-switching cost problem, that has the following regret bounds*

- *For any time interval $I \subseteq [T]$, the regret of the algorithm w.r.t. expert 0 is at most 1.*

- *The regret of the algorithm w.r.t. expert 1 is at most $O(\sqrt{DT\log(TD)})$*

Theorem 10 is the main building block for proving Theorem 1. We first use it to combine two algorithms. Namely, given two algorithms $\mathcal{A}_0$ and $\mathcal{A}_1$ for the N-experts D-switching cost problem, we use Theorem 10 to combine them into a single algorithm the preserves regret bounds of $\mathcal{A}_0$ and $\mathcal{A}_1$, plus additional quantities, as in Theorem 10. Then, we use this basic combining procedure to combine many algorithm, deriving Theorem 1.

A sketch of Theorem 10's proof. In this section we highlight the main ideas in the proof of Theorem 10, and make a few simplifying assumption to help the presentation, including that $D = 1$. At time step $t \geq 1$, the player chooses $x_t \in [0,1]$ and suffers a loss of $\ell_t^{on} = \ell_t(0)(1 - $

Figure 1: $g(x)$ for $Z = 10^{-8}$ and $\tau = 10000$
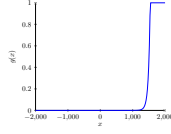
$x_t) + \ell_t(1)x_t + D|x_t - x_{t-1}|$. The algorithm has two parameters: $\tau \geq 1$ and $Z > 0$. Define $\tilde{g}$ to be the solution of the differential equation

$$8\tilde{g}'(x) = \frac{1}{\tau}x\tilde{g}(x) + Z, \quad \tilde{g}(0) = 0 . \tag{4}$$

Define $U = U_{\tau,Z} := \tilde{g}^{-1}(1)$ ($\tilde{g}$ is strictly increasing and unbounded, so U is well defined, and we later show that $U \leq \sqrt{16\tau \log\left(\frac{1}{Z}\right)}$). Denote the projection to $[a, b]$ by $\Pi_{[a,b]}(z) = \begin{cases} a & z \leq a \\ z & a \leq z \leq b \\ b & z \geq b \end{cases}$.

Finally, we define $g(x) = \Pi_{[0,1]}[\tilde{g}(x)]$, which ensures that $0 \leq g(x) \leq 1$ and that for $x \leq 0$ we have $g(x) = 0$ and for $x \geq U$ we have $g(x) = 1$. (see Figure 1 for the plot of $g(x)$). The following algorithm is a simplification of Algorithm 2 that achieves the regret bounds of Theorem 10.

Algorithm 1 Two experts (with parameters τ and Z)

1: Set $x_t = 0$
2: **for** $t = 1, 2, \ldots$ **do**
3:     Predict $g(x_t)$
4:     Let $b_t = l_t(0) - l_t(1)$ and update $x_{t+1} = \left(1 - \frac{1}{\tau}\right)x_t + b_t$
5: **end for**

We next elaborate on the proof. To highlight the main ideas, we will sketch the proof of just for two special cases: (1) showing that the regret to the second expert for the time interval $[1, \tau]$ is at most $\sqrt{64\tau \log\left(\frac{1}{Z}\right)} + 2 + \tau Z$, and (2) For the time interval $[T]$, the regret of w.r.t. the first expert is at most TZ. We need the following helpful notation. Let $G(x) = \int_0^x g(s)ds$, denote $\Phi_t = G(x_t)$ and let $I : \mathbb{R} \to \{0, 1\}$ be the indicator function of the segment $[-2, U+2]$. We consider the change

in the value of Φ_t,

$$
\begin{aligned}
\Phi_{t+1} - \Phi_t &= \int_{x_t}^{x_{t+1}} g(s)ds = \int_{x_t}^{x_t - \frac{1}{\tau}x_t + b_t} g(s)ds \\
&\leq g(x_t)\left(-\frac{1}{\tau}x_t + b_t\right) + \frac{1}{2}\left(-\frac{1}{\tau}x_t + b_t\right)^2 \max_{s\in[x_t, x_t - \tau^{-1}x_t + b_t]}|g'(s)| \\
&\leq g(x_t)\left(-\frac{1}{\tau}x_t + b_t\right) + \frac{1}{2}4 \max_{s\in[x_t, x_t - \tau^{-1}x_t + b_t]}|g'(s)| \\
&= g(x_t)\left(-\frac{1}{\tau}x_t + b_t\right) - 2\max_{s\in[x_t, x_t - \tau^{-1}x_t + b_t]}|g'(s)| + 4\max_{s\in[x_t, x_t - \tau^{-1}x_t + b_t]}|g'(s)| \\
&\leq g(x_t)\left(-\frac{1}{\tau}x_t + b_t\right) - 2\frac{|g(x_t) - g(x_{t+1})|}{|-\frac{1}{\tau}x_t + b_t|} + 4\max_{s\in[x_t, x_t - \tau^{-1}x_t + b_t]}|g'(s)| \\
&\leq g(x_t)\left(-\frac{1}{\tau}x_t + b_t\right) - |g(x_t) - g(x_{t+1})| + 4\max_{s\in[x_t, x_t - \tau^{-1}x_t + b_t]}|g'(s)| \\
&\leq g(x_t)\left(-\frac{1}{\tau}x_t + b_t\right) - |g(x_t) - g(x_{t+1})| + \frac{1}{\tau}x_t g(x_t)I(x_t) + Z
\end{aligned}
$$

Here, the first inequality follows from the fact that for every piece-wise differential function $f : [a,b] \to \mathbb{R}$ we have $\int_a^b f(x)dx \leq f(a)(b-a) + \frac{1}{2}(b-a)^2 \max_{\xi\in[a,b]}|f'(\xi)|$. The second and forth inequalities follows from the fact that by a simple induction, $|x_t| \leq \tau$ and hence $\left|-\frac{x_t}{\tau} + b_t\right| \leq 2$. The third inequality follows from the mean value Theorem. As for the last inequality, since g is a solution of the ODE from equation (4), and is constant outside $[0, U]$ we have $4g'(x_t) \leq \frac{1}{2}\left[\frac{1}{\tau}x_t g(x_t)I(x_t) + Z\right]$. We later show that g is smooth enough so that the inequality remains valid, up to a factor of 4, on the entire interval $[x_t, x_t - \tau^{-1}x_t + b_t]$. Namely, $4g'(x_t) \leq \frac{1}{\tau}x_t g(x_t)I(x_t) + Z$. Summing from $t = 1$ to $t = T$, rearranging, and using the fact that $\Phi_1 = 0$ we get,

$$
-\sum_{t=1}^{T} g(x_t)b_t + \sum_{t=1}^{T}|g(x_t) - g(x_{t+1})| \leq -\Phi_{T+1} + \sum_{t=1}^{T}\frac{1}{\tau}x_t g(x_t)(I(x_t) - 1) + TZ \quad (5)
$$

Now, denote the loss of the algorithm up to time t by L_t^{on}, and by L_t^i the loss of the expert i. We have

$$
\begin{aligned}
L_t^{on} &= \sum_{t'=1}^{t} g(x_{t'})l_{t'}(1) + \sum_{t'=1}^{t}(1 - g(x_{t'}))l_{t'}(0) + \sum_{t'=1}^{t}|g(x_{t'}) - g(x_{t'+1})| \\
&= L_t^0 + \left(-\sum_{t'=1}^{t} g(x_{t'})b_{t'} + \sum_{t'=1}^{t}|g(x_{t'}) - g(x_{t'+1})|\right) \\
&\leq L_t^0 - \Phi_{t+1} + \sum_{t'=1}^{t}\frac{1}{\tau}x_{t'}g(x_{t'})(I(x_{t'}) - 1) + tZ, \quad (6)
\end{aligned}
$$

where the inequality follows from (5). When considering the entire interval $[T]$, we can derive the following regret. Since $\Phi_{T+1} \geq 0$ and $x_{t'}g(x_t)(I(x_{t'}) - 1) \leq 0$ we have $L_T^{on} \leq L_T^0 + TZ$, which proves that the regret of the first expert over $[T]$. We now prove the regret bound for the second expert, in our special case, i.e., to bound $L_T^{on} - L_T^1$. Recall that $g(z) = 1$ any $z \geq U$, $0 \leq g(z) \leq 1$

11

for $z \in [0, U]$, and $g(z) = 0$ for any $z \leq 0$. Therefore, we have $\Phi_{\tau+1} = \int_0^{x_{\tau+1}} g(s)ds \geq x_{\tau+1} - U$. Also, $x_t g(x_t)(1 - I(x_t)) \geq x_t - U - 2$, since for $z \geq U$ we have $g(z) = 1$. Therefore, if we denote $b_0 := 0$, we have

$$
\begin{aligned}
\Phi_{\tau+1} + \sum_{t=1}^{\tau} \frac{1}{\tau} x_t g(x_t)(1 - I(x_t)) \quad &\geq \quad (x_{\tau+1} - U) + \sum_{t=1}^{\tau} \frac{1}{\tau}(x_t - U - 2) \\
&= \quad \sum_{j=0}^{\tau} \left(1 - \frac{1}{\tau}\right)^{\tau-j} b_j + \sum_{t=1}^{\tau} \frac{1}{\tau} \sum_{j=0}^{t-1} \left(1 - \frac{1}{\tau}\right)^{t-1-j} b_j - 2(U+1) \\
&= \quad \sum_{j=0}^{\tau} \left(1 - \frac{1}{\tau}\right)^{\tau-j} b_j + \frac{1}{\tau} \sum_{j=0}^{\tau-1} \sum_{t=j+1}^{\tau} \left(1 - \frac{1}{\tau}\right)^{t-1-j} b_j - 2(U+1) \\
&= \quad \sum_{j=0}^{\tau} \left(1 - \frac{1}{\tau}\right)^{\tau-j} b_j + \frac{1}{\tau} \sum_{j=0}^{\tau} \frac{1 - \left(1 - \frac{1}{\tau}\right)^{\tau-j}}{\frac{1}{\tau}} b_j - 2(U+1) \\
&= \quad \sum_{j=0}^{\tau} b_j - 2U = L_\tau^0 - L_\tau^1 - 2(U+1)
\end{aligned}
$$

The above shows that $L_\tau^1 + 2(U+1) \geq L_\tau^0 - \Phi_{\tau+1} + \sum_{t=1}^{\tau} \frac{1}{\tau} x_t g(x_t)(I(x_t) - 1)$. By equation (6), $L_\tau^1 + 2(U+1) + \tau Z \geq L_\tau^{on}$. The proof is concluded by showing that $U \leq \sqrt{16\tau \log\left(\frac{1}{Z}\right)}$.

3.1. Metrical Tasks Systems

We will derive our results for MTSs from the following variant of Theorem 1

Theorem 12 *There is a procedure that given as input experts algorithms $\mathcal{A}_{\text{base}}, \mathcal{A}_0, \dots, \mathcal{A}_{\log_2(T)}$, combines them into a single algorithm $\mathcal{A}$ such that:*

1. *On any interval I of length $\frac{T}{2^{u+1}} \leq |I|\frac{T}{2^u}$, the regret of $\mathcal{A}$ w.r.t. $\mathcal{A}_u$ is $O\left(D\sqrt{|I|\log(T)}\right)$*

2. *The regret of $\mathcal{A}$ w.r.t. $\mathcal{A}_{\text{base}}$ is D*

Furthermore, if the original algorithms are efficient, then so is $\mathcal{A}$. More precisely, at each round the procedure is given as input the loss of each algorithm in that round, and an indication which algorithms made switches. Then, the procedure specifies one of the experts algorithms, and $\mathcal{A}$ chooses its action. The computational overhead of the procedure is $O\left(\log(T)\right)$.[2]

The combination of the various algorithms is done as follows. We start with $\mathcal{A}_{\text{base}}$ and combine it, using the two experts algorithm, with $\mathcal{A}_0$. This yields an algorithm $\mathcal{B}_0$ with essentially no regret w.r.t. $\mathcal{A}_{\text{base}}$ and small regret w.r.t. $\mathcal{A}_0$. Then, we continue doing so, and at step i, we combine $\mathcal{B}_i$ with $\mathcal{A}_{i+1}$ to obtain $\mathcal{B}_{i+1}$.

To prove Corollary 4 (which also implies Corollaries 5 and 8), we take $\mathcal{A}_{\text{base}}$ be an algorithm with competitive ratio of α, and $\mathcal{A}_0, \dots, \mathcal{A}_{\log_2 T}$ to be the algorithm from Corollary 3. Likewise, we set the switching costs to be the diameter of the underlying metric space. We run algorithm $\mathcal{A}_u$ with parameter $\tau_u = 2^{-u}T$ and set $Z_u = 1/T$.

---

2. Assuming that arithmetic operations, exponentiation, computing the error function, and sampling uniformly from $[0, 1]$ cost $O(1)$.

Paging. To prove Corollary 7 we take $\mathcal{A}_{\text{base}}$ to be some paging algorithm with competitive ratio $2H_k$ (say, the marking algorithm Fiat et al. (1991)). For $0 \leq u \leq \log_2 T$, $\mathcal{A}_u$ is obtained by sequentially applying an algorithm with time horizon $2^{-u}T$, that enjoys the regret bound from Theorem 6, and has per round running time $\text{poly}(n, \log(T))$. Likewise, we set the switching costs to be k. The corollary follows easily from Theorem 12 when I has the form $[(j-1)2^{-u}T+1, j2^{-u}T]$. Otherwise, by a standard chaining argument (e.g., Daniely et al. (2015)), we can decompose I into a disjoint union on segments $I_0, \ldots, I_l$ of the form $[(j-1)2^{-u}T+1, j2^{-u}T]$ such that $|I_i| \leq |I|2^{-\frac{i-1}{2}}$. Summing the regret bound on the different segments yields a geometric sequence, establishing the corollary.

In section 8 we show that the multiplicative weights algorithm, applied to the paging problem, enjoys the regret bounds from Theorem 6. Furthermore, we show that despite the exponential number of experts, it can be implemented efficiently. The basic idea is the following. The MW algorithm maintains a positive weight for each expert, and at each step predicts the probability distribution obtained by dividing each weight by the sum of the weights. We note that in paging, due to the special structure of the loss vectors, after the page requests $i_1, \ldots, i_t$, the weight of the expert corresponding to the cache $A \in \binom{[N]}{k}$ is $\prod_{t'=1}^{t} e^{\eta \mathbb{1}[i_{t'} \in A]}$ where $\eta = \sqrt{\frac{\log\left(\binom{n}{k}\right)}{kT}}$. This special structure enables the use of dynamic programming in order to implement the MW algorithm efficiently.

4. Future Directions

We presented a regret minimization methodology to classic online computation problems, which can interpolate between a static benchmark to a dynamic one. An important building block was developing an expert algorithm that is strongly adaptive even in the presence of switching costs. As elaborated below, our work leaves many open directions.

Maybe the most interesting and fruitful direction is to find other online problems that can fall into our framework and for which we can develop computationally efficient online algorithms. We believe that our results can be extended the many other online problems such as competitive data structures (see Blum et al. (2002) for results in this direction), buffering, scheduling, load balancing, etc. Likewise, our expert results can be extended to other settings such as partial information models[3], strategic environments, non-oblivious adversaries, etc. In our approach there is a design issue of selecting the benchmark to consider. We have taken probably the most obvious benchmark, an online algorithm that does not change its state, however, one can consider other benchmarks which are more problem specific. In addition to this grand challenge, there are also open problems regarding our specific analysis.

While most of our regret bounds are tight, some regret bounds have certain gaps, which would be interesting to overcome. For MTS, the regret bound for the algorithm that ensures optimal competitive ratio (Corollary 4) is worst by a factor of $\sqrt{D}$ compared to the algorithm without this guarantee (Corollary 3). We wonder if this gap can be reconciled. For paging, we conjecture that there are efficient algorithms with regret bound $O(\sqrt{kT \log(N)})$, which is better than our bound of $O(k\sqrt{T \log(N)})$. For k-server, we conjecture that it is NP-hard to efficiently achieve a regret bound of $O(\sqrt{\text{poly}(k, D)T})$. On the other hand, we conjecture that there is a constant $c > 1$ such

---

3. We remark that in the classic bandit setting strong adaptivity is impossible, even without switching costs Daniely et al. (2015).

that there are efficient algorithms whose cost is at most c times the costs of the best fixed-locations strategy, plus $O(\sqrt{\mathrm{poly}(k,D)T})$. We note that since there are $(2k-1)$-competitive deterministic and $O\left(\log^2(k)\right)$-competitive randomized algorithms for k-sever, our conjecture is true if c is not a constant, but rather $(2k-1)$ or $O\left(\log^2(k)\right)$. In addition to improving our bounds, obtaining simpler algorithms with similar guarantees would be of great interest. Specifically, for paging, k-server and MTS, the obtained algorithms are somewhat cumbersome and obtained by combining many algorithms (especially when we insist on ensuring optimal competitive ratio).

Finally, regarding our expert algorithm and the tightness of the regret bounds in the case of switching cost 1, Theorem 1 is tight up to a constant factor. Yet, for general $D \geq 1$, there is a larger gap. While our regret bound is $O\left(\sqrt{D|I|\log{(NT)}}\right)$, the best known lower bound (that is obtained by combining Theorem 2 with Geulen et al. (2010)) is $\Omega\left(\sqrt{D|I|\log{(N)}} + \sqrt{|I|\log{(NT)}}\right)$. Another interesting direction is to find a simpler algorithm and analysis for Theorem 1.

5. Online Linear Optimization with Switching Cost

5.1. Notations and conventions.

For simplicity, we assume throughout that $T = 2^K$ and that the switching cost $D \geq 1$.[4]

The letter τ will be used to denote the length of time intervals $I \subset [T]$, and therefore by convention τ is an integer that is greater 0.

We denote by $\Delta(N)$ the simplex over a set N elements. Unless otherwise stated, the norm over points in the simplex is the total variation norm, i.e., $\|\cdot\| = \frac{\|\cdot\|_1}{2}$. For $x \in \mathbb{R}$ we denote $x_+ = \max(0, x)$ and for $x \in \mathbb{R}^d$ we let $x_+ = ((x_1)_+, \ldots, (x_d)_+)$.

For a piece-wise differentiable function $f : (a, b) \to \mathbb{R}$ and $x \in (a, b)$ we use the convention that $|f'(x)|$ denotes the maximum over the right and left derivatives of f at x. For integrable function $f : [a, b] \to \mathbb{R}$ we denote $\int_b^a f(x)dx := -\int_a^b f(x)dx$. For a segment $I = [a, b]$ we let $\Pi_I(x)$ be the projection on I, namely,

$$\Pi_I(x) = \begin{cases} b & x > b \\ x & a \leq x \leq b \\ a & x < a \end{cases}$$

5.2. From N-experts D-switching cost to online linear optimization with switching cost

Our first step is to reduce the N-experts D-switching cost problem (abbreviated $\mathrm{EXP}(N, D)$) to the problem of online linear optimization over $\Delta(N)$ with total variation switching cost and parameter $D \geq 1$ (abbreviated $\mathrm{OLO}(N, D)$). The latter problem is defined as follows. The game is played for T time steps, such that at each time step $t = 1, 2, \ldots, T$

- Nature chooses a loss $l_t \in [0, 1]^N$

- The learner choose an action $x_t \in \Delta(N)$

- The player suffers a loss of $\langle l_t, x_t \rangle + D\|x_t - x_{t-1}\|$ (or just $\langle l_t, x_t \rangle$ if $t = 1$)

---

4. This is justified since our bounds in the case that $D < 1$ are the same as the case $D = 1$, and the results for $D < 1$ can be obtained by a simple reduction to the case $D = 1$.

We assume that nature is oblivious (i.e., the loss sequence was chosen before the game started) and that the learner is deterministic and its action at step t depends only on the past losses $l_1, \ldots, l_{t-1}$. The notion of regret is define in Section 2.

As we will show in this section, $\mathrm{OLO}(N, D)$ is essentially equivalent to $\mathrm{EXP}(N, D)$. Namely, we will show that any algorithm for $\mathrm{OLO}(N, D)$ can be transformed to an algorithm for $\mathrm{EXP}(N, D)$ such that for every loss sequence, the loss (expected loss in the case of $\mathrm{EXP}(N, D)$) remains the same. Likewise, any algorithm for $\mathrm{EXP}(N, D)$ can be transformed to an algorithm for $\mathrm{OLO}(N, D)$ such that for every loss sequence, the loss does not grow.

We start by reducing $\mathrm{EXP}(N, D)$ to $\mathrm{OLO}(N, D)$. Let $\mathcal{A}$ be an algorithm for $\mathrm{OLO}(N, D)$. We will explain how to transform it to a learning algorithm $\mathcal{A}'$ for $\mathrm{EXP}(N, D)$. Let $\mathcal{L} = \{l_1, \ldots, l_T\} \subset [0, 1]^N$ be a loss sequence, and let $x_1, \ldots, x_T \in \Delta(N)$ be the distributions over actions of $\mathcal{A}$ on that sequence. In Lemma 15 we show that for every pair of consecutive distributions over actions x_{t-1}, x_t, there is a transition probability matrix $p_t(i|j)$, $i, j \in [N]$ such that if X_{t-1} is distributed according to x_{t-1} and X_t is generated from X_{t-1} based on p_t, then X_t is distributed according to x_t and moreover, $\Pr(X_t \neq X_{t-1}) = \|x_t - x_{t-1}\|$. Given this, we can construct an algorithm $\mathcal{A}'$ such that the expert chosen at round 1 is a random expert X_1 distributed according to x_1, and for each $t > 1$, X_t is generated from X_{t-1} according to the transition probability matrix p_t. The above discussion shows that at each step t the loss of $\mathcal{A}'$ in $EXP(N, D)$ is

$$\mathbb{E} l_t(X_t) + D \Pr(X_t \neq X_{t-1}) = \langle l_t, x_t \rangle + D\|x_t - x_{t-1}\| \,,$$

which is exactly the loss of $\mathcal{A}$ at the same step in $OLO(N, D)$.

We first prove the following simple fact.

Lemma 13 *For every $z, z' \in \Delta(N)$, $\|z - z'\| = \sum_{j=1}^{N}(z'_j - z_j)_+ = \sum_{j=1}^{N}(z_j - z'_j)_+$, and $\frac{(z-z')_+}{\|z-z'\|}$ is a distribution.*

Proof Since for any distribution the probabilities sum to 1, we have that $\sum_{i=1}^{N} z'_i - z_i = 0$. By splitting the last sum to positive and negative summands we conclude that

$$\sum_{i=1}^{N}(z'_i - z_i)_+ = \sum_{i=1}^{N}(z_i - z'_i)_+ \,.$$

Hence, considering the total variation norm, we have,

$$\|z - z'\| = \frac{\sum_{i=1}^{N}|z'_i - z_i|}{2} = \frac{\sum_{i=1}^{N}((z'_i - z_i)_+) + \sum_{i=1}^{N}(z_i - z'_i)_+}{2} = \sum_{j=1}^{N}(z'_j - z_j)_+$$

The fact that $\frac{(z-z')_+}{\|z-z'\|}$ is a distribution follows from the fact that $\sum_{j=1}^{N} \frac{(z_j - z'_j)_+}{\|z-z'\|} = 1$. ∎

We first define the transition probability matrix.

Definition 14 *For distributions $z, z' \in \Delta(N)$ we define*

$$p_{z,z'}(i, j) = \min\{z_i, z'_j\}\mathbb{1}[i = j] + \frac{((z_i - z'_i)_+)((z'_j - z_j)_+)}{\|z - z'\|}$$

15

It remains to state and prove the lemma for the joint distribution.

Lemma 15 *For any $z, z' \in \Delta(N)$, $p_{z,z'}$ is a distribution on $[N] \times [N]$. Furthermore, if (X, X') is a random variable distributed according to $p_{z,z'}$ then*

1. *X is distributed according to z and X' is distributed according to z', and*

2. *$\Pr(X \neq X') = \|z - z'\|$*

Note that if $z = z'$ then we have that $p_{z,z'}(i, j) = z_i 1[i = j]$, namely, the support is only the pairs (i, i).

Remark 16 *Suppose that X is distributed according to z and we want to generate X' that distributed according to z' and $\Pr(X \neq X') = \|z - z'\|$. We can sample as follows. Assume that $X = i$, then with probability $\frac{(z_i - z'_i)_+}{z_i}$ sample X' from the distribution $\frac{(z'-z)_+}{\|z-z'\|}$ and w.p. $1 - \frac{(z_i - z'_i)_+}{z_i}$ set $X' := X$.*

Proof (of Lemma 15) Assume that $z \neq z'$. By Fact 13, we have that for every $i \in [N]$,

$$
\begin{aligned}
\sum_{j=1}^{N} p_{z,z'}(i, j) &= \min\{z_i, z'_i\} + \frac{1}{\|z - z'\|}((z_i - z'_i)_+) \sum_{j=1}^{N}(z'_j - z_j)_+ \\
&= \min\{z_i, z'_i\} + \frac{1}{\|z - z'\|}((z_i - z'_i)_+)\|z - z'\| \qquad (7) \\
&= \min\{z_i, z'_i\} + ((z_i - z'_i)_+) = z_i
\end{aligned}
$$

Similarly, for every $j \in [N]$,

$$
\sum_{i=1}^{N} p_{z,z'}(i, j) = z_j, \qquad (8)
$$

and,

$$
\sum_{i=1}^{N} \sum_{j=1}^{N} p_{z,z'}(i, j) = \sum_{i=1}^{N} z_i = 1 . \qquad (9)
$$

Equations (7), (8) and (9) show that $p_{z,z'}$ is a distribution whose marginals are z and z'. It remains to prove that if (X, X') is distributed according to $p_{z,z'}$ then $\Pr(X \neq X') = \|z - z'\|$. Indeed, again by Fact 13,

$$
\Pr(X \neq X') = 1 - \sum_{i=1}^{N} \min\{z_i, z'_i\} = \sum_{i=1}^{N} z_i - \min\{z_i, z'_i\} = \sum_{i=1}^{N}(z_i - z'_i)_+ = \|z - z'\|
$$

$\square$

We now show how an $\mathcal{A}'$ algorithm for $\mathrm{EXP}(N, D)$ can be transformed to an algorithm $\mathcal{A}$ for $\mathrm{OLO}(N, D)$ such that on any loss sequence, the loss of $\mathcal{A}$ in $\mathrm{OLO}(N, D)$ is the same as the loss of $\mathcal{A}'$ in $EXP(N, D)$. To this end, let $\mathcal{L}$ be a loss sequence, and let $X_t \in [N]$ be the expert chosen by $\mathcal{A}'$ when it runs on $\mathcal{L}$. Let $x_t \in \Delta(N)$ be the distribution of X_t. The algorithm $\mathcal{A}$ will simply play x_t at round t. The loss of $\mathcal{A}$ in $\mathrm{OLO}(N, D)$ at step t is

$$
\langle l_t, x_t \rangle + D\|x_t - x_{t-1}\|,
$$

while the loss of $\mathcal{A}'$ in $\mathrm{EXP}(N, D)$ is

$$\langle l_t, x_t \rangle + D \Pr (X_t \neq X_{t-1})$$

which is equal to the loss of $\mathcal{A}$ by Lemma 15.

5.3. An algorithm for two experts

Consider the $\mathrm{OLO}(N, D)$ problem with $N = 2$. It will be convenient to use $x \in [0, 1]$ to denote the probability distribution $(x, 1 - x)$. Concretely, the game is defined as follows. At each step $t \geq 1$,

- The adversary chooses $l_t = (l_t(0), l_t(1)) \in [0, 1]^2$.

- The player chooses $z_t \in [0, 1]$ and suffers a loss of

$$l_t = l_t(0)(1 - z_t) + l_t(1)z_t + D|z_t - z_{t-1}|$$

(for $t = 1$, assume that $z_0 := 0$)

- l_t is revealed to the player.

Consider the following algorithm. Let $\tau, D \geq 1$ and $Z > 0$. Let $er\!f(x) = \int_0^x \exp\left(-\frac{s^2}{2}\right) ds$.[5]
Define $\tilde{g} = \tilde{g}_{\tau, Z}$ as follows

$$\tilde{g}(x) = \sqrt{\frac{\tau}{8}} Z er\!f\left(\frac{x}{\sqrt{8\tau}}\right) \exp\left(\frac{x^2}{16\tau}\right) \tag{10}$$

We note that $\tilde{g}$ is a solution of the differential equation

$$8\tilde{g}'(x) = \frac{1}{\tau} x \tilde{g}(x) + Z . \tag{11}$$

Define $U = U_{\tau, Z} := \tilde{g}^{-1}(1)$. Also, define $g = g_{\tau, Z}$ as

$$g(x) = \Pi_{[0,1]} [\tilde{g}(x)] = \begin{cases} 0 & x \leq 0 \\ \tilde{g}(x) & 0 \leq x \leq U_{\tau, Z} \\ 1 & x \geq U_{\tau, Z} \end{cases}$$

Algorithm 2 Two experts (with parameters τ, D and Z)

1: Set $x_t = 0$
2: **for** $t = 1, 2, \ldots$ **do**
3:    **if** $D \log\left(\frac{1}{Z}\right) \leq \frac{\tau}{64}$ **then**
4:       Predict $g(x_t)$
5:    **else**
6:       Predict 0
7:    **end if**
8:    Let $b_t = \frac{l_t(0) - l_t(1)}{\sqrt{D}}$
9:    Update $x_{t+1} = \Pi_{[-2, U+2]}\left[\left(1 - \frac{1}{\tau}\right) x_t + b_t\right]$
10: **end for**

---

5. Note that this definition is slightly different form the standard error-function that is defined as $\frac{2}{\sqrt{\pi}} \int_0^x \exp\left(-s^2\right) ds$.

To parse the following theorem, we note that when we will apply it, we will take Z to be very small (say, T^{-10}). Likewise, τ can ba any number $0 \leq \tau \leq T$, potentially much smaller than T. In these settings, the theorem shows that the algorithm has very small regret with respect to expert 0. Namely the regret is $o(1)$ even on segments that are much larger than τ. Remarkably, the algorithm is able to achieve this while preserving almost optimal $\tilde{O}\left(\sqrt{\tau}\right)$ regret w.r.t. expert 1, but only on segments of length τ.

Theorem 17 *Suppose $Z \leq \frac{1}{e}$. Algorithm 2 guarantees,*

- *For every time interval I, the regret w.r.t. expert 0 is at most* $\min\{\sqrt{D}TZ, \sqrt{16D\tau \log\left(\frac{1}{Z}\right)} + 2\sqrt{D} + \sqrt{D}|I|Z\}$

- *For every time interval I of length $\leq \tau$, the regret w.r.t. expert 1 is at most* $\sqrt{64D\tau \log\left(\frac{1}{Z}\right)} + 4\sqrt{D} + \sqrt{D}\tau Z$

5.3.1. PROOF OF THEOREM 17

Properties of $g_{\tau,Z}$ We first prove some properties of the function $\tilde{g}$.

Lemma 18   *The function $\tilde{g}(x)$ has the following properties:*

1. *$\tilde{g}(x)$ is strictly increasing odd function.*

2. *$\tilde{g}(x)$ is convex in $[0, \infty)$.*

3. *For $\tau \geq 8e$ and $Z \leq \frac{1}{e}$ we have $U_{\tau,Z} \leq \sqrt{16\tau \log\left(\frac{1}{Z}\right)}$, where $U_{\tau,Z} := \tilde{g}^{-1}(1)$*

Proof Part 1 follows immediately form equation (10). For part 2, note that $\tilde{g}'(x) \geq 0$ and that

$$8\tilde{g}''(x) = \frac{1}{\tau}\tilde{g}(x) + \frac{1}{\tau}x\tilde{g}'(x) . \tag{12}$$

Hence, $\tilde{g}''$ is non-negative in $[0, \infty)$ and therefore $\tilde{g}$ is convex. For Part 3, we have

$$
\begin{aligned}
\tilde{g}\left(\sqrt{16\tau \log\left(\frac{1}{Z}\right)}\right) &= \sqrt{\frac{\tau}{8}}Z erf\left(\sqrt{2\log\left(\frac{1}{Z}\right)}\right)\frac{1}{Z} \\
&\geq \sqrt{\frac{\tau}{8}}erf(1) \\
&= \sqrt{\frac{\tau}{8}}\int_0^1 e^{-\frac{x^2}{2}}dx \\
&\geq \sqrt{\frac{\tau}{8}}e^{-\frac{1}{2}} = \frac{\sqrt{\tau}}{\sqrt{8e}} > 1
\end{aligned}
$$

The lemma follows since $\tilde{g}$ is increasing. $\square$

Lemma 19 *Suppose* $\log\left(\frac{1}{Z}\right) \leq \frac{\tau}{16}$, $Z \leq \frac{1}{e}$ *and* $\tau \geq 8e$. *For every segment* $I \subset \mathbb{R}$ *of length* ≤ 2 *and every* $x \in I$ *we have*[6]

$$4 \max_{s \in I} |g'(s)| \leq \frac{1}{\tau} x g(x) + Z \tag{13}$$

Proof Let $I = [a, b]$. The function $\frac{1}{\tau} x g(x) + Z$ is non decreasing (since it is constant outside $[0, U_{\tau,Z}]$ and is the derivative of the convex function $8\tilde{g}$ inside $[0, U_{\tau,Z}]$). Therefore, it is enough to show that

$$4 \max_{s \in [a,b]} |g'(s)| \leq \frac{1}{\tau} a g(a) + Z \tag{14}$$

We first claim that we can restrict to the case that $I \subset [0, U_{\tau,Z}]$. Indeed, if $a > U_{\tau,Z}$ or $b < 0$, then the l.h.s. is 0 and the claim holds, so we can assume that $a \leq U_{\tau,Z}$ and $b \geq 0$. Next, if we replace b with $\min\{b, U_{\tau,Z}\}$, both the l.h.s. and r.h.s. of (14) remains unchanged, as $g'(s) = 0$ for $s > U_{\tau,Z}$. Therefore, we can also assume that $b \leq U_{\tau,Z}$. Likewise, if we replace a with $\max\{a, 0\}$, both the l.h.s. and r.h.s. of (14) remains unchanged, as $g'(s) = 0$ for $s < 0$ and $\frac{1}{\tau} a g(a) + Z = Z$ for $a \leq 0$.

By Gronwall's inequality for ODE, if $g''(x) \leq \frac{1}{4} g'(x)$, we will have that for all $s \in I$,

$$\begin{aligned}
g'(s) &\leq g'(a) \exp\left(\frac{s-a}{4}\right) \\
&\leq g'(a)\sqrt{e} \\
&= \frac{\sqrt{e}}{8\tau} a g(a) + \frac{\sqrt{e}}{8} Z \\
&\leq \frac{1}{4\tau} a g(a) + \frac{1}{4} Z
\end{aligned}$$

Therefore, it is sufficient to show that for all $x \in I$ we have

$$g''(x) \leq \frac{1}{4} g'(x) . \tag{15}$$

By (12), since $I \subset [0, U_{\tau,Z}]$ and since $\log\left(\frac{1}{Z}\right) \leq \frac{\tau}{16}$, by Lemma 18, we have $U_{\tau,Z} \leq \sqrt{16\tau \log\left(\frac{1}{Z}\right)} \leq \tau$. Since $x \leq U_{\tau,Z} \leq \tau$ which implies,

$$\begin{aligned}
g''(x) &\leq \frac{1}{8\tau} g(x) + \frac{x}{8\tau} g'(x) \\
&\leq \frac{1}{8\tau} g(x) + \frac{1}{8} g'(x)
\end{aligned}$$

It therefore remains to show that $\frac{1}{8\tau} g(x) \leq \frac{1}{8} g'(x)$. By (11) it is equivalent to

$$\frac{g(x)}{\tau} \leq \frac{x g(x)}{8\tau} + \frac{Z}{8}$$

---

6. when s is 0 or $U_{\tau,Z}$ (i.e, when g is not differentiable), $|g'(s)|$ stands for the maximum of the absolute values of the left and right derivatives.

For $x \geq 8$ the inequality is clear. For $x \leq 8$ we will show that $\frac{g(x)}{\tau} \leq \frac{Z}{8}$. Indeed, for such x, since $\tau \geq 8e$, we have

$$
\begin{aligned}
g(x) &\leq \sqrt{\frac{\tau}{8}} Z \left(\int_0^{\frac{8}{\sqrt{8\tau}}} \exp\left(-\frac{s^2}{2}\right) ds \right) \exp\left(\frac{4}{\tau}\right) \\
&\leq \sqrt{\frac{\tau}{8}} Z \frac{8}{\sqrt{8\tau}} \exp\left(\frac{4}{\tau}\right) \\
&\leq \sqrt{\frac{\tau}{8}} Z \frac{8}{\sqrt{8 \cdot 8e}} \exp\left(\frac{4}{8e}\right) \\
&\leq \sqrt{\frac{\tau}{8}} Z \sqrt{e} \leq \frac{\tau Z}{8}
\end{aligned}
$$

$\square$

Removing the projection and restricting to a finite horizon The next step is to show that in order to prove Theorem 17, we can consider a version of algorithm 2 with finite time horizon and no projection. Concretely, consider the following algorithm:

Algorithm 3 Two experts without projection and with bounded horizon

---

Parameters: Initial $x_1 \in [-2, U+2], T, \tau, Z, D$.

1: **for** $t = 1, 2, \ldots, T$ **do**
2:    **if** $D \log\left(\frac{1}{Z}\right) \leq \frac{\tau}{64}$ **then**
3:       Predict $g(x_t)$
4:    **else**
5:       Predict 0
6:    **end if**
7:    Let $b_t = \frac{l_t(0) - l_t(1)}{\sqrt{D}}$
8:    Update $x_{t+1} = \left(1 - \frac{1}{\tau}\right) x_t + b_t$
9: **end for**

For expert $i \in \{0, 1\}$ and an interval I we denote by $R^{i,alg2}_{I,\tau,D,Z}$ the worst case regret of algorithm 2 on the interval I when running with parameters τ, D, Z. Likewise, we denote by $R^{i,alg3}_{T,\tau,D,Z}$ the worst case regret (over all possible loss sequences and initial points x_1) of algorithm 3 when running with parameters τ, D, Z, T.

Lemma 20 $R^{i,alg2}_{I,\tau,D,Z} \leq R^{i,alg3}_{|I|,\tau,D,Z}$

Proof Denote $T = |I|$ and $I = \{K+1, \ldots, K+T\}$. We claim that there exists a sequence $\mathcal{L} = \{l_1, l_2, \ldots\} \subset [0,1]^2$ of losses such that

- Algorithm 2 suffers a regret of $R^{i,alg2}_{I,\tau,D,Z}$ on the segment I when running on $\mathcal{L}$

- If we let $b_1 = \frac{l_1(0) - l_1(1)}{\sqrt{D}}, b_2 = \frac{l_2(0) - l_2(1)}{\sqrt{D}}, \ldots$ and let $x_1, x_2, \ldots$ be the actions that algorithm 2 chose, then $\forall t \in I, \tilde{x}_{t+1} := \left(1 - \frac{1}{\tau}\right) x_t + b_t \in [-2, U+2]$.

This will prove the lemma, because in that case the actions, and therefore the regret of algorithm 3 on the sequence $l_{K+1}, \ldots, l_{K+T}$ with initial point x_{K+1} is identical to algorithm 2 on I. In particular, $R_{I,\tau,D,Z}^{i,alg2} \leq R_{|I|,\tau,D,Z}^{i,alg3}$.

Assume toward a contradiction that there is no such $\mathcal{L}$. Choose $\mathcal{L}$ among all sequences causing a regret of $R_{I,\tau,D,Z}^{i,alg2}$, in a way that the first step $t \in I$ for which $\tilde{x}_{t+1} \notin [-2, U+2]$ is as large as possible. Assume that $\tilde{x}_{t+1} > U + 2$ (a similar argument holds if $\tilde{x}_{t+1} < -2$). This implies that $l_t(0) > l_t(1)$ and $g(x_t) = 1$. Now, suppose we generate a new sequence by decreasing $l_t(0)$ in a way that we would have $\tilde{x}_{t+1} = U + 2$. This will not change the loss of the algorithm in step t (as $g(x_t) = 1$) and won't change x_{t+1}, and therefore won't change the actions of the algorithm and its losses in the remaining steps. As for the experts, this will only improve the loss of expert 0. Therefore, the regret will not decrease. This contradicts the minimality of $\mathcal{L}$. $\qquad\square$

Completing the proof We first discuss the case that $D \log\left(\frac{1}{Z}\right) > \frac{\tau}{64}$, which is much simpler. In that case the algorithm will simply choose the expert 0 at each round. Hence, the regret w.r.t. expert 0 will be zero. Likewise, since the algorithm does not move at all, the regret w.r.t. expert 1 on an interval of length τ is at most τ. Since

$$\tau = \sqrt{\tau}\sqrt{\tau} < \sqrt{\tau}\sqrt{64D \log\left(\frac{1}{Z}\right)}$$

we are done. Therefore, for the rest of the proof, we assume that $D \log\left(\frac{1}{Z}\right) \leq \frac{\tau}{64}$. In this case, and by lemma 20, it is enough to prove the following lemma:

Lemma 21 *Suppose $D \log\left(\frac{1}{Z}\right) \leq \frac{\tau}{64}$ and $Z \leq \frac{1}{e}$. Then, for any initial point $x_1 \in [-2, U+2]$, algorithm 3 guarantees the following,*

- *The regret w.r.t. expert 0 is at most* $\min\{\sqrt{D}TZ, \sqrt{16D\tau \log\left(\frac{1}{Z}\right)} + 2\sqrt{D} + \sqrt{D}TZ\}$

- *If $T \leq \tau$, the regret w.r.t. expert 1 is at most* $\sqrt{64D\tau \log\left(\frac{1}{Z}\right)} + \sqrt{D}4 + \sqrt{D}\tau Z$

The proof is almost identical to the one sketched in Section 3 and is deferred to the Appendix.

5.4. Combining Algorithms

5.4.1. COMBINING TWO ALGORITHMS

We next describe a variant of algorithm 2 to combine two algorithms $\mathcal{A}_0$ and $\mathcal{A}_1$ for linear optimization over $\Delta(N)$ with switching costs D. The resulting algorithm will have tiny regret w.r.t. to $\mathcal{A}_0$ and small regret w.r.t. $\mathcal{A}_1$. To describe it and analyze it, we will use the following terminology. We will say that and algorithm is M-*slow* if for every loss sequence, the distance between two consecutive actions is at most M.

Algorithm 4 Two algorithm combiner (with parameters τ, Z, M and D)

Parameters: $\frac{M}{D}$-slow algorithms $\mathcal{A}_0, \mathcal{A}_1$ for online linear optimization over $\Delta(N)$ with switching costs D.

1: Set $x_t = 0$, $g = g_{\tau,Z}$, $U = U_{\tau,Z}$
2: **for** $t = 1, 2, \ldots$ **do**
3:     Let z_t^0, z_t^1 be the actions of $\mathcal{A}_0, \mathcal{A}_1$
4:     **if** $\tau \geq 64D \log\left(\frac{1}{Z}\right)$ **then**
5:         Predict $g(x_t)z_t^0 + (1 - g(x_t))z_t^1$
6:     **else**
7:         Predict z_t^0
8:     **end if**
9:     Obtain loss vector l_t and let $\tilde{l}_t(i) = \frac{\langle l_t, z_t^i \rangle + D\|z_t^i - z_{t+1}^i\|}{M+1}$ be the scaled loss of algorithm i
10:     Let $b_t = \frac{\tilde{l}_t(0) - \tilde{l}_t(1)}{\sqrt{D}}$
11:     Update $x_{t+1} = \Pi_{[-2, U+2]}\left[\left(1 - \frac{1}{\tau}\right)x_t + b_t\right]$
12: **end for**

Using Theorem 17 we conclude that

Theorem 22 *Suppose $Z \leq \frac{1}{e}$. Algorithm 4 guarantees:*

- *For every time interval I, the regret w.r.t. $\mathcal{A}_0$, i.e., $(M+1)\sum_{t \in I} \tilde{l}_t(0)$, is at most*

$$\left((M+1)\sqrt{D}\right)\min\{TZ, \sqrt{16\tau \log\left(\frac{1}{Z}\right)} + 2 + |I|Z\}$$

- *For every time interval I of length $\leq \tau$, the regret w.r.t. $\mathcal{A}_1$, i.e., $(M+1)\sum_{t \in I} \tilde{l}_t(1)$, is at most*

$$\left((M+1)\sqrt{D}\right)\left(\sqrt{64\tau \log\left(\frac{1}{Z}\right)} + 4 + \tau Z\right)$$

Proof The loss of the combined algorithm at time t is

$$
\begin{aligned}
l_t^{\text{comb}} &= \langle l_t, g(x_t)z_t^0 + (1 - g(x_t))z_t^1 \rangle \\
&\quad + D\|g(x_t)z_t^0 + (1 - g(x_t))z_t^1 - g(x_{t+1})z_{t+1}^0 + (1 - g(x_{t+1}))z_{t+1}^1\| \\
&\leq \langle l_t, g(x_t)z_t^0 + (1 - g(x_t))z_t^1 \rangle + D\|g(x_t)(z_t^0 - z_{t+1}^0) + (1 - g(x_t))(z_t^1 - z_{t+1}^1)\| \\
&\quad + D\|(g(x_t) - g(x_{t+1}))z_{t+1}^0 + ((1 - g(x_t)) - (1 - g(x_{t+1})))z_{t+1}^1\| \\
&\leq \langle l_t, g(x_t)z_t^0 + (1 - g(x_t))z_t^1 \rangle + g(x_t)D\|z_t^0 - z_{t+1}^0\| + (1 - g(x_t))D\|z_t^1 - z_{t+1}^1\| \\
&\quad + D|g(x_t) - g(x_{t+1})| \\
&= (M+1)\left(g(x_t)\tilde{l}_t(0) + (1 - g(x_t))\tilde{l}_t(1)\right) + D|g(x_t) - g(x_{t+1})| \\
&\leq (M+1)\left(g(x_t)\tilde{l}_t(0) + (1 - g(x_t))\tilde{l}_t(1) + D|g(x_t) - g(x_{t+1})|\right)
\end{aligned}
$$

By Theorem 17, for every time interval I we have

$$\sum_{t \in I} g(x_t)\tilde{l}_t(0) + (1 - g(x_t))\tilde{l}_t(1) + D|g(x_t) - g(x_{t+1})| \le$$

$$\sum_{t \in I} \tilde{l}_t(0) + \min\left\{\sqrt{D}|I|Z, \sqrt{16D\tau \log\left(\frac{1}{Z}\right)} + 2\sqrt{D} + \sqrt{D}|I|Z\right\}$$

Therefore we have that

$$\sum_{t \in I} l_t^{\text{comb}} - l_t^{\mathcal{A}_0} \le \left((M+1)\sqrt{D}\right) \min\{TZ, \sqrt{16\tau \log\left(\frac{1}{Z}\right)} + 2 + |I|Z\}$$

Finally, if $|I| \le \tau$ then

$$\sum_{t \in I} g(x_t)\tilde{l}_t(0) + (1 - g(x_t))\tilde{l}_t(1) + D|g(x_t) - g(x_{t+1})| \le \sum_{t \in I} \tilde{l}_t(1) + \sqrt{64D\tau \log\left(\frac{1}{Z}\right)} + 4\sqrt{D} + \sqrt{D}\tau Z$$

which implies that

$$\sum_{t \in I} l_t^{\text{comb}} - l_t^{\mathcal{A}_1} \le \left((M+1)\sqrt{D}\right)\left(\sqrt{64\tau \log\left(\frac{1}{Z}\right)} + 4 + \tau Z\right)$$

$\square$

We next bound the slowness of the combined algorithm.

Lemma 23 *Suppose $Z \le \frac{1}{e}$. The combined algorithm is $\left(\frac{M}{D} + \sqrt{\frac{\log(\frac{1}{Z})}{4\tau D}} + \frac{Z}{8\sqrt{D}}\right)$-slow if $\tau \ge 64D \log\left(\frac{1}{Z}\right)$ and $\frac{M}{D}$-slow otherwise.*

Proof Clearly, if $\tau < 64D \log\left(\frac{1}{Z}\right)$, we select z_t^0 and the bound follows from the bound on $\mathcal{A}_0$.

Assume that $\tau \ge 64D \log\left(\frac{1}{Z}\right)$. The movement of the combined algorithm at each step is bounded by a convex combination of the movements of $\mathcal{A}_0$ and $\mathcal{A}_1$ plus $|g(x_t) - g(x_{t+1})|$. More specifically

$$g(x_{t+1})z_{t+1}^0 - g(x_t)z_t^0 = g(x_{t+1})(z_{t+1}^0 - z_t^0) + (g(x_{t+1}) - g(x_t))z_t^0$$

Similarly

$$(1 - g(x_{t+1}))z_{t+1}^1 - (1 - g(x_t))z_t^1 = (z_{t+1}^1 - z_t^1)(1 - g(x_{t+1})) + (g(x_t) - g(x_{t+1}))z_t^1$$

Since we have that algorithms $\mathcal{A}_b$ are M/D-slow we have $z_{t+1}^b - z_t^b \le M/D$. What remains is to bound

$$|g(x_t) - g(x_{t+1})| \le |x_t - x_{t+1}| \max_\xi |g'(\xi)| \le \frac{\max_\xi |g'(\xi)|}{\sqrt{D}},$$

where we used the fact that $|x_t - x_{t+1}| \le |b_t| \le 1/\sqrt{D}$. Since g' is nondecreasing in $[0, U_{\tau,Z}]$ and is 0 outside, we have

$$\max_\xi |g'(\xi)| = g'(U_{\tau,z}) = \frac{U_{\tau,z}g(U_{\tau,z})}{8\tau} + \frac{Z}{8} = \frac{U_{\tau,z}}{8\tau} + \frac{Z}{8}$$

and the lemma follows, since by Lemma 18 we have $U_{\tau,Z} \le \sqrt{16\tau \ln(1/Z)}$. $\blacksquare$

5.4.2. COMBINING MANY ALGORITHMS

For simplicity, let's assume that $T = 2^K$. Let $\mathcal{A}_{\text{base}}, \mathcal{A}_0, \dots, \mathcal{A}_{K-1}$ be $\frac{1}{D}$-slow algorithms. We next explain how one can sequentially use algorithm 4 to combine these algorithms into a single algorithm $\mathcal{A}$, while preserving some of their guarantees. Namely, on every interval I of length $|I| \in [2^{-u-1}T, \leq 2^{-u}T]$, $\mathcal{A}$ will have small loss w.r.t. $\mathcal{A}_u$. In addition, on the entire segment $[T]$, $\mathcal{A}$ will have essentially no loss w.r.t. $\mathcal{A}_{\text{base}}$. We note that in our application, the role of the $\mathcal{A}_u$'s will be to ensure strong adaptivity. On the other hand, the role of $\mathcal{A}_{\text{base}}$ will be to ensure competitive ratio, and it can be easily omitted from when this is not needed.

We will build algorithms $\mathcal{B}_{-1}, \mathcal{B}_0, \dots, \mathcal{B}_{K-1}$ where $\mathcal{B}_{-1} = \mathcal{A}_{\text{base}}$, and for $u \geq 0$ the algorithm $\mathcal{B}_u$ is obtained from $\mathcal{B}_{u-1}$ by combining it with $\mathcal{A}_u$ using algorithm 4 (where $\mathcal{B}_{u-1}$ plays the role of $\mathcal{A}_0$ from algorithm 4 and $\mathcal{A}_u$ the role of $\mathcal{A}_1$). Finally, we will take $\mathcal{A} = \mathcal{B}_{K-1}$. The parameter Z will be the same among all the applications of algorithm 4, but we will assume that $Z \leq \frac{1}{e}$. The parameter τ will be set to $2^{-u}T$ when we combine $\mathcal{A}_u$. Lastly, the slowness bound on $\mathcal{A}_u$ will be $\frac{1}{D}$, while the slowness bound on $\mathcal{B}_{u-1}$ is the one implied by Lemma 23. Namely,

$$\max\left(\frac{1}{D} + \sum_{i=0}^{u-1} \sqrt{\frac{\log\left(\frac{1}{Z}\right)}{4(2^{-i}T)D}} + \frac{Z}{8\sqrt{D}} \, , \, 1 \right)$$

Theorem 24 *Assume $Z \leq \frac{1}{e}$*

1. *On each interval I of length $2^{-u-1} \leq |I| \leq 2^{-u}T$, the regret of $\mathcal{A}$ w.r.t. $\mathcal{A}_u$ is*
$$O\left(\sqrt{D|I| \log\left(\frac{1}{Z}\right)} + \sqrt{D} \log(|I|)|I|Z \right)$$

2. *The regret of $\mathcal{A}$ w.r.t. $\mathcal{A}_{\text{base}}$ is $2\sqrt{D}T \log(T)Z$*

Proof (sketch) As with previous proofs, we can assume w.l.o.g. that $T \geq D \log_2(T)$. We first claim that for every u, $\mathcal{B}_u$ is $\frac{2}{D}$-slow, and hence we can use Theorem 22 with $M = 2$. Indeed, $\mathcal{B}_0 = \mathcal{A}_0$ is $\frac{1}{D}$-slow by assumption. Now, when we go from $\mathcal{B}_{u-1}$ to $\mathcal{B}_u$, by Lemma 23, the slowness grows by

$$\sqrt{\frac{\log\left(\frac{1}{Z}\right)}{4(2^{-u}T)D}} + \frac{Z}{8\sqrt{D}}$$

as long as $2^{-u}T \geq 64D \log\left(\frac{1}{Z}\right)$, and by 0 after that. It follows that the total growth is bounded by

$$\sum_{u=0}^{\left\lfloor \log_2\left(\frac{T}{64D \log\left(\frac{1}{Z}\right)} \right) \right\rfloor} \sqrt{\frac{\log\left(\frac{1}{Z}\right)}{42^{-u}TD}} + \frac{Z}{8\sqrt{D}} \quad \leq \quad \sqrt{\frac{\log\left(\frac{1}{Z}\right)}{4TD}} \sqrt{2}^{\log_2\left(\frac{T}{64D \log\left(\frac{1}{Z}\right)} \right)} \frac{\sqrt{2}}{\sqrt{2}-1} + \frac{\log_2(T)Z}{8\sqrt{D}}$$

$$= \quad \sqrt{\frac{\log\left(\frac{1}{Z}\right)}{4TD}} \sqrt{\frac{T}{64D \log\left(\frac{1}{Z}\right)}} \frac{\sqrt{2}}{\sqrt{2}-1} + \frac{\log_2(T)}{8T\sqrt{D}}$$

$$\leq \quad \frac{1}{\sqrt{256D}}4 + \frac{1}{8D} \leq \frac{1}{2D}$$

Now, let I be an interval of length $\leq 2^{-u}T =: \tau$ for some u. By Theorem 22, the regret of $\mathcal{B}_u$ w.r.t. $\mathcal{A}_u$ on that interval is, up to a universal multiplicative constant, at most

$$\sqrt{D\tau \log\left(\frac{1}{Z}\right)} + \sqrt{D}\tau Z$$

24

Now, in order to go from $\mathcal{B}_u$ to $\mathcal{B}_K$, we sequentially combine the algorithms $\mathcal{A}_{u+1}, \ldots, \mathcal{A}_K$. By Theorem 22, up to a universal multiplicative constant, this adds to the regret on the given segment at most

$$\sum_{r=1}^{K-u-1} \left(\sqrt{D \cdot 2^{-r} \tau \log\left(\frac{1}{Z}\right)} + \sqrt{D}\tau Z \right) \leq \sqrt{D}\left(\sum_{r=1}^{\infty} 2^{-\frac{r}{2}} \right) \sqrt{\tau \log\left(\frac{1}{Z}\right)} + \sqrt{D}(K-u-1)\tau Z$$

$$\leq 20\sqrt{D\tau \log\left(\frac{1}{Z}\right)} + \sqrt{D}\log(\tau)\tau Z$$

This proves the first part of the Theorem. The proof of the second part is similar. □

Proof (of Theorem 12) The proof follows from Theorem 24 with $Z = \frac{1}{2T\log(T)}$. Indeed, in the case that $D = 1$, the requirement of being $\frac{1}{D}$-slow always holds. The general case follows by a simple scaling argument. As for running time, in the case that the algorithms choose a specific expert (rather than a distribution on the expert), in order to apply algorithm 4 all is needed is the loss of the chosen expert and an indication weather the algorithms made switches. □

6. Experts and Metrical Tasks Systems

6.1. Algorithms

In this section we prove Theorem 1. We will use Theorem 24 where the basic algorithms are the fixed share algorithm Herbster and Warmuth (1998). We extend its analysis to handle switching costs.

Algorithm 5 Fixed Share Herbster and Warmuth (1998)

Parameters: τ, D

1: Set $\eta = \sqrt{\frac{\log(N\tau)}{D\tau}}$

2: Set $z_1 = \left(\frac{1}{N}, \ldots, \frac{1}{N}\right)$

3: **for** $t = 1, 2, \ldots,$ **do**

4:     Predict z_t

5:     **if** $\tau \geq 16D\log(N\tau)$ **then**

6:         Update $z_{t+1}(i) = \frac{z_t(i)e^{-\eta l_t(i)} + \frac{1}{N\tau}}{\sum_{j=1}^{N}\left(z_t(j)e^{-\eta l_t(j)} + \frac{1}{N\tau}\right)}$

7:     **end if**

8: **end for**

We first bound the rate of change in the action distribution which bounds the slowness of the algorithm.

Lemma 25 *Let $\eta > 0, \tau \geq \frac{2}{\eta}$ and $l_1, \ldots, l_N \in [0,1]$. Let $z \in \Delta(N)$ and define $z'(i) = \frac{e^{-\eta l_i} z(i) + \frac{1}{N\tau}}{\sum_{j=1}^{N} e^{-\eta l_j} z(j) + \frac{1}{N\tau}}$. Then $\|z - z'\| \leq \eta$*

Proof Denote $\tilde{z}(i) = z(i)e^{-\eta l_i} + \frac{1}{N\tau}$. For the proof it will be more convenient to use norm L_1 and recall that $\|z\|_1 = 2\|z\|$. We have

$$\|z' - z\|_1 \leq \|z' - \tilde{z}\|_1 + \|\tilde{z} - z\|_1. \tag{16}$$

We bound the contribution of each term independently. For the second term we have,

$$
\begin{aligned}
\|\tilde{z} - z\|_1 &\leq \sum_{i=1}^{N} |z(i)(1 - e^{-\eta l_i})| + \frac{1}{N\tau} \\
&= \frac{1}{\tau} + \sum_{i=1}^{N} z(i)(1 - e^{-\eta l_i}) \\
&\leq \frac{1}{\tau} + \sum_{i=1}^{N} z(i)\eta l_i \\
&\leq \frac{1}{\tau} + \sum_{i=1}^{N} z(i)\eta = \frac{1}{\tau} + \eta \quad\quad (17)
\end{aligned}
$$

For the first term we have,

$$
\|\tilde{z} - z'\|_1 = \left\| \tilde{z} - \frac{\tilde{z}}{\|\tilde{z}\|_1} \right\|_1 = \left| 1 - \frac{1}{\|\tilde{z}\|_1} \right| \cdot \|\tilde{z}\|_1 = \left| \|\tilde{z}\|_1 - 1 \right|
$$

To bound this we have,

$$
\|\tilde{z}\|_1 \geq \frac{1}{\tau} + \|z\|_1 e^{-\eta} = \frac{1}{\tau} + e^{-\eta} \geq \frac{1}{\tau} + 1 - \eta
$$

and also

$$
\|\tilde{z}\|_1 \leq \|z\|_1 + \frac{1}{\tau} = 1 + \frac{1}{\tau}
$$

and we have

$$
\left| \|\tilde{z}\|_1 - 1 \right| \leq \max\{\frac{1}{\tau}, \frac{1}{\tau} - \eta\} = \frac{1}{\tau}
$$

Combining with equations (16) and (17), and since $\tau \geq \frac{2}{\eta}$, we conclude that

$$
\|z' - z\| = \frac{\|z' - z\|_1}{2} \leq \frac{1}{\tau} + \frac{\eta}{2} \leq \eta
$$

∎

An immediate corollary is bounding the slowness of the algorithm.

Corollary 26 *Algorithm 5 is $\sqrt{\frac{\log(N\tau)}{D\tau}}$-slow when $\tau \geq 16D \log(N\tau)$ and 0-slow otherwise.*

We can now derive the regret bounds.

Theorem 27 *On every time interval of length $\leq \tau$, the regret (including switching costs) of MW^1 is bounded by $\sqrt{16D\tau \log(N\tau)}$*

Proof If $\tau < 16D \log (N\tau)$, the algorithm makes no moves, so its regret is bounded by $\tau = \sqrt{\tau}\sqrt{\tau} \leq \sqrt{3D\tau} \leq \sqrt{16D\tau \log (N\tau)}$. We can therefore assume that $\tau \geq 16D \log (N\tau)$. Hazan and Koren (2016) showed that in this case the regret of the algorithm, excluding switching costs is at most $\frac{2 \log(N\tau)}{\eta} + \eta\tau$. By Lemma 25 (the fact that $\tau \geq \frac{2}{\eta}$ follows from the assumption that $\tau \geq 16D \log (N\tau)$) the switching cost in each round is bounded by $D\eta$. Hence, the regret is at most $\frac{2 \log(N\tau)}{\eta} + \eta\tau + D\eta\tau \leq \frac{2 \log(N\tau)}{\eta} + 2D\eta\tau = \sqrt{16D \log (N\tau)\tau}$. ∎

We are now ready to prove Theorem 1

Proof (of Theorem 1) Let $\mathcal{A}_0, \ldots, \mathcal{A}_{K-1}$ be instances of algorithm 5 with parameters $\tau = 2^{-0}T, \tau = 2^{-1}T, \ldots, \tau = 2^{-K+1}T$. Let $\mathcal{A}$ be the algorithm obtained by Theorem 24 with $Z = \frac{1}{2T \log(T)}$. By Theorems 24 and 27 we have that the regret of $\mathcal{A}$ on every interval I is

$$O\left(\sqrt{D|I|\log(NT)} + \sqrt{D|I|\log(|I|)} + \sqrt{D}\right) = O\left(\sqrt{D|I|\log(NT)}\right)$$

□

6.2. A lower bound

In this section we will prove Theorem 2 that shows that the bound in Theorem 1 is optimal up to a constant factor. We start by showing how the adversary can generate sequences of guaranteed high loss.

Lemma 28 *Let $\mathcal{A}$ be an algorithm for linear optimization over $\Delta(N)$ for $N = 2$. Suppose $\mathcal{A}$ have a regret bound of M on the interval $[T]$ and assume that $T \geq 4M$. There is a sequence of loses $l_1, \ldots, l_T \in \left\{(0,1),(1,0),\left(\frac{1}{2},\frac{1}{2}\right)\right\}$ for which the loss of the algorithm is at least $\frac{1}{2}T + M2^{-4M}$*

Proof Since we have two actions, our distribution over actions at time t would be $(1 - z_t, z_t)$. Assume that the initial choice of the algorithm is $z_1 \leq \frac{1}{2}$ (a similar argument holds when $z_1 \geq \frac{1}{2}$). Let $a = M2^{-4M}$ and assume toward a contradiction that the is no such sequence. Namely, $\mathcal{A}$ is guaranteed to have a loss of at most $\frac{1}{2}T + a$ on every sequence. We denote by L_t the loss of the algorithm at the end of round t, and define the *gain* of the algorithm at time t as $G_t = \frac{t}{2} - L_t$. We claim that we have $z_{t+1} \leq \frac{1}{2} + G_t + a$. Indeed, otherwise, the adversary can cause the gain to be $< -a$ at the next step $t + 1$ by choosing the loss vector $(0, 1)$. It can also keep the gain $< -a$ by repeatedly choosing the loss vectors $\left(\frac{1}{2}, \frac{1}{2}\right)$.

Consider now the action of the algorithm when the loss vectors are always $(1, 0)$. We claim that $z_t \leq \frac{1}{2} + 2^{t-1}a$. We will prove this by induction. For $t = 1$ it follows from our assumption that $z_1 \leq \frac{1}{2}$. Assume that this is the case for all $t' < t$. We have that the loss of the algorithm before the step t, is at least

$$\left(\sum_{t'=1}^{t-1} \frac{1}{2} - 2^{t'-1}a\right) = \frac{t-1}{2} - (2^{t-1} - 1)a \tag{18}$$

Therefore, $G_{t-1} \leq (2^{t-1} - 1)a$. It follows that

$$z_t \leq \frac{1}{2} + G_{t-1} + a \leq \frac{1}{2} + (2^{t-1} - 1)a + a = \frac{1}{2} + 2^{t-1}a$$

Now, using equation (18) again, the regret at time t w.r.t. to the second expert is at least $\frac{t}{2} - 2^t a \leq M$. Taking $t = 4M$, it follows that $a \geq M2^{-4M}$ ∎

We will now use the sequences guaranteed by the above lemma to show a lower bound on the regret.

Theorem 29 *For every algorithm for online linear optimization over $\Delta(N)$ that runs for T iterations, there is a segment I on which the regret is $\Omega\left(\sqrt{|I|\log(NT)}\right)$.*

Proof The known regret lower bounds guarantee that the regret is $\Omega\left(\sqrt{|I|\log(N)}\right)$. We first note that it is enough to show that that for some interval I the regret is $\Omega\left(\sqrt{|I|\log(T)}\right)$. Hence, we will have a regret lower bound of

$$\Omega\left(\max\left\{\sqrt{|I|\log(N)}, \sqrt{|I|\log(T)}\right\}\right) = \Omega\left(\sqrt{|I|\log(N)} + \sqrt{|I|\log(T)}\right)$$
$$= \Omega\left(\sqrt{|I|\log(N) + |I|\log(T)}\right)$$
$$= \Omega\left(\sqrt{|I|\log(NT)}\right)$$

It is now left to show that for some interval I the regret is $\Omega\left(\sqrt{|I|\log(T)}\right)$. We will show that this lower bound holds already in easier problem of linear optimization over $\Delta(N)$, where $N = 2$. Indeed, suppose toward a contradiction that there is an algorithm $\mathcal{A}$ whose regret is $\leq \frac{1}{100}\sqrt{|I|\log_2(T)}$ on every interval $I \subset \{1, \ldots, T\}$.

Partition $[T]$ into intervals of size $\log_2(T)$. By Lemma 28, there is a sequence of losses that all come from the set $\left\{(0,1), (1,0), \left(\frac{1}{2}, \frac{1}{2}\right)\right\}$, such that the loss on every interval is at least $\frac{\log_2(T)}{2} + M2^{-4M}$ for $M = \frac{\log_2(T)}{100}$. The loss over the entire interval $[T]$ is therefore at least $\frac{T}{2} + \frac{T}{\log_2(T)}M2^{-4M} = \frac{T}{2} + \frac{T}{100}2^{-\frac{\log_2(T)}{25}} = \frac{T}{2} + \frac{T^{\frac{24}{25}}}{100}$. The regret is therefore at least $\frac{T^{\frac{24}{25}}}{100}$, contradicting the assumption that it is at most $\frac{1}{100}\sqrt{T\log(T)}$. ∎

7. k-Server

Recall that in the k-median problem we are given a metric space (X, d), and the goal is to find a set $A \subset X$ of size k that minimizes $\text{VAL}_d(A) = \sum_{x \in X} d(x, A)$, where $d(x, A) = \min_{a \in A} d(x, a)$.

Lemma 30 *Assume that there is an efficient[7] algorithm for the k-server problem with regret bound of $\text{poly}(n)T^{1-\mu}$ for some $\mu > 0$. Then, for every $\epsilon > 0$ there is an efficient $(2 + \epsilon)$-approximation algorithm for the k-median problem.*

Proof Let $\mathcal{A}$ be an efficient algorithm for k-server with regret bound of $f(n)T^{1-\mu}$ for some polynomially bounded function f. Let (X, d) be an n-points metric space that is an instance for the k-median problem. Let $A^* \subset X$ be a set of k points that minimizes VAL_d and denote $\text{OPT} = \text{OPT}_d = \text{VAL}_d(A^*)$.

We first note that we can assume w.l.o.g. that $\text{OPT} \geq 1$ and that $D := \max_{x,y} d(x, y) \leq 3$. Indeed, we can compute a number $\text{OPT} \leq \alpha \leq 3\text{OPT}$ using known approximation algorithms

---

7. Namely, one that runs in each step in time polynomial in n, T and the bit-representation of the underlying metric-space.

(e.g., using Byrka et al. (2015)). Now, instead of working with the original metric d, we can work with the metric $d'(x,x') = \frac{3\min(d(x,x'),\alpha)}{\alpha}$. Clearly, its diameter is bounded by 3. Moreover, we claim that $\mathrm{OPT}_{d'} = \frac{3\mathrm{OPT}_d}{\alpha}$ and therefore $\mathrm{OPT}_{d'} \geq 1$ and any $(2+\epsilon)$-approximation w.r.t. d' is also a $(2+\epsilon)$-approximation w.r.t d. Indeed, since $d' \leq \frac{3}{\alpha}d$ we have $\mathrm{OPT}_{d'} \leq \frac{3\mathrm{OPT}_d}{\alpha}$. On the other hand, we claim that it cannot be the case that $\mathrm{VAL}_{d'}(A) < \frac{3\mathrm{OPT}_d}{\alpha}$. Indeed, in that case we must have $d'(x,A) < \frac{3\mathrm{OPT}_d}{\alpha} \leq 3$ for all $x \in X$ in which case $d'(x,A) = \frac{3}{\alpha}d(x,A)$. Hence, $\mathrm{VAL}_{d'}(A) = \frac{3}{\alpha}\mathrm{VAL}_d(A) \geq \frac{3\mathrm{OPT}_d}{\alpha}$. A contradiction.

Suppose now that we run the k-server algorithm such that at each round we choose a point $x \in X$ uniformly at random. If we run the algorithm for $T = \left(\frac{f(n)n}{\epsilon}\right)^{\frac{1}{\mu}}$ rounds, we are guaranteed to have expected regret $\leq \frac{\epsilon}{n}T$. Denote by A_t the location of servers at the beginning of round t. The expected cost is at least $\sum_{t=1}^{T}\mathbb{E}_{x\sim X}d(x,A_t)$. On the other hand, if $A^* \subset X$ is an optimal solution to the k-medians problem, the expected loss of the corresponding k-server strategy is $\sum_{t=1}^{T}\mathbb{E}_{x\sim X}2d(x,A^*)$. Since the regret is bounded by $\frac{\epsilon}{n}T$, we have $\sum_{t=1}^{T}\mathbb{E}_{x\sim X}d(x,A_t) - 2d(x,A^*) \leq \frac{\epsilon}{n}T$. In particular, if we choose at random one of the A_t's as a solution to the k-median problem, we get a solution with expected cost at most $\mathbb{E}_{x\sim X}2d(x,A^*) + \epsilon = 2\mathrm{OPT} + \epsilon \leq (2+\epsilon)\mathrm{OPT}$. ∎

8. Paging

We first recall the paging problem. To simplify the presentation a bit, we consider a version where both the losses and the movements cost are divided by 2. At each step t the player has to choose a set $A_t \in \binom{[N]}{k}$. Then, nature chooses an element $i_t \in [N]$, and the player loses 1 if $i_t \notin A_t$. In addition, the player suffers a switching cost of $\frac{k}{2}$. Consider first the multiplicative weight algorithm Littlestone and Warmuth (1994) for the N-expert problem

---
Algorithm 6 MW

Parameters: T, D

1: Set $\eta = \sqrt{\frac{\log(N)}{2DT}}$
2: Set $z_1 = \left(\frac{1}{N}, \ldots, \frac{1}{N}\right) \in \Delta(N)$
3: **for** $t = 1, 2, \ldots, T$ **do**
4:     Predict z_t
5:     Update $z_{t+1}(i) = \frac{z_t(i)e^{-\eta l_t(i)}}{\sum_{j=1}^{N} z_t(j)e^{-\eta l_t(j)}}$
6: **end for**

Theorem 31 *The regret (including switching costs) of* MW *is bounded by* $\sqrt{8DT\log(N)}$

Proof It is known (e.g., Cesa-Bianchi and Lugosi (2006), page 15) the regret of the algorithm, excluding switching costs is at most $\frac{\log(N)}{\eta} + \eta T$. By Lemma 25 (and taking τ to ∞) the switching cost in each round is bounded by ηD. Hence, the regret is at most $\frac{\log(N)}{\eta} + 2D\eta T = \sqrt{8D\log(N)T}$. □

By Theorem 31, the MW algorithm has a regret of $\sqrt{8\frac{k}{2}T\log\binom{N}{k}} \leq k\sqrt{4T\log(N)}$ in the paging problem. However, a naive implementation of the algorithm will result with an algorithm

whose running time is exponential in k. As we explain next, a more careful implementation will result with an efficient algorithm. Let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N) \in (0, \infty)^N$. For $A \subset [N]$ denote $\pi_{\boldsymbol{\alpha}}(A) = \prod_{i \in A} \alpha_i$ and $\Psi_{N,k}^{\boldsymbol{\alpha}} = \sum_{A' \in \binom{[N]}{k}} \pi_{\boldsymbol{\alpha}}(A)$. Let $p_{\boldsymbol{\alpha}}^k$ be the distribution function on $\binom{[N]}{k}$ defined by $p_{\boldsymbol{\alpha}}^k(A) = \frac{\pi_{\boldsymbol{\alpha}}(A)}{\Psi_{N,k}^{\boldsymbol{\alpha}}}$. The multiplicative weights algorithm for paging can be described as follows:

Algorithm 7 Multiplicative weights for paging prediction

1: Set $\eta = \sqrt{\frac{\log\binom{N}{k}}{kT}}$
2: Set $\boldsymbol{\alpha}^1 = (1, \ldots, 1) \in \mathbb{R}^N$
3: **for** $t = 1, 2, \ldots, T$ **do**
4:     Choose a set $A_t \sim p_{\boldsymbol{\alpha}^t}^k$ such that $\Pr(A_t \neq A_{t-1}) = \|p_{\boldsymbol{\alpha}^t}^k - p_{\boldsymbol{\alpha}^{t-1}}^k\|$ (when $t > 1$)
5:     Update $\alpha_{i_t}^{t+1} = e^{\eta} \alpha_{i_t}^t$ and $\alpha_i^{t+1} = \alpha_i^t$ for all $i \neq i_t$.
6: **end for**

We next remark on the computational complexity of some sampling and calculation procedures related to the family of distributions $\{p_{\boldsymbol{\alpha}}^k\}_{\boldsymbol{\alpha}}$. The last point shows that algorithm 7 can be implemented efficiently, which implies Theorem 6. In the sequel, efficient means polynomial in the the description length of $\boldsymbol{\alpha}$. We denote $\boldsymbol{\alpha}_- = (\alpha_1, \ldots, \alpha_{N-1})$

- **Computing.** In order to efficiently compute $p_{\boldsymbol{\alpha}}^k(A)$ it is enough to efficiently compute $\Psi_{N,k}^{\boldsymbol{\alpha}}$. This computation is straight forward when $k = 1$ or $k = N$. When $1 < k < N$ we have

$$
\begin{aligned}
\Psi_{N,k}^{\boldsymbol{\alpha}} &= \sum_{A \in \binom{[N]}{k},\, N \in A} \pi_{\boldsymbol{\alpha}}(A) + \sum_{A \in \binom{[N]}{k},\, N \notin A} \pi_{\boldsymbol{\alpha}}(A) \\
&= \alpha_N \sum_{A \in \binom{[N-1]}{k-1}} \pi_{\boldsymbol{\alpha}}(A) + \sum_{A \in \binom{[N-1]}{k}} \pi_{\boldsymbol{\alpha}}(A) \\
&= \alpha_N \Psi_{N-1,k-1}^{\boldsymbol{\alpha}_-} + \Psi_{N-1,k}^{\boldsymbol{\alpha}_-}
\end{aligned}
$$

hence, $\Psi_{N,k}^{\boldsymbol{\alpha}}$ can be efficiently calculated using dynamic programming.

- **Sampling.** Suppose that $A \in \binom{[N]}{k}$ is sampled form $p_{\boldsymbol{\alpha}}^k$. We have

$$
\Pr(N \notin A) = \frac{\Psi_{N-1,k}^{\boldsymbol{\alpha}_-}}{\Psi_{N,k}^{\boldsymbol{\alpha}}} \qquad \Pr(N \in A) = \frac{\alpha_N \Psi_{N-1,k-1}^{\boldsymbol{\alpha}_-}}{\Psi_{N,k}^{\boldsymbol{\alpha}}} \tag{19}
$$

Also for $A' \in \binom{[N-1]}{k-1}$ we have

$$
\Pr(A \setminus \{N\} = A' \mid N \in A) = \frac{p_{\boldsymbol{\alpha}}^k(A' \cup \{N\})}{\Pr(N \in A)} = \frac{\frac{\alpha_N \pi_{\boldsymbol{\alpha}_-}(A')}{\Psi_{N,k}^{\boldsymbol{\alpha}}}}{\frac{\alpha_N \Psi_{N-1,k-1}^{\boldsymbol{\alpha}_-}}{\Psi_{N,k}^{\boldsymbol{\alpha}}}} = \frac{\pi_{\boldsymbol{\alpha}_-}(A')}{\Psi_{N-1,k-1}^{\boldsymbol{\alpha}_-}} = p_{\boldsymbol{\alpha}_-}^{k-1}(A') .
$$

Likewise, for $A' \in \binom{[N-1]}{k}$ we have

$$
\Pr(A \setminus \{N\} = A' \mid N \notin A) = \frac{p_{\boldsymbol{\alpha}}^k(A')}{\Pr(N \notin A)} = \frac{\frac{\pi_{\boldsymbol{\alpha}}(A')}{\Psi_{N,k}^{\boldsymbol{\alpha}}}}{\frac{\Psi_{N-1,k}^{\boldsymbol{\alpha}_-}}{\Psi_{N,k}^{\boldsymbol{\alpha}}}} = \frac{\pi_{\boldsymbol{\alpha}}(A')}{\Psi_{N-1,k}^{\boldsymbol{\alpha}_-}} = p_{\boldsymbol{\alpha}_-}^k(A') .
$$

Hence, in order to efficiently sample from $p_{\boldsymbol{\alpha}}^k$ we can first choose whether to include N in A according to equation (19). Then (recursively) sample $A \setminus \{N\}$ from $p_{\boldsymbol{\alpha}_-}^{k-1}$ in the case that $N \in A$ and from $p_{\boldsymbol{\alpha}_-}^k$ if $N \notin A$.

- **Transitioning with minimal switching costs.** Let $\boldsymbol{\alpha} \in (0, \infty)^N$ and $\boldsymbol{\alpha}' = (\alpha_1, \ldots, \alpha_{N-1}, \alpha_N + \delta)$. Suppose that $A \sim p_{\boldsymbol{\alpha}}^k$ and we want to efficiently generate $A' \sim p_{\boldsymbol{\alpha}'}^k$ such that $\Pr(A \neq A') = \|p_{\boldsymbol{\alpha}}^k - p_{\boldsymbol{\alpha}'}^k\|$. According to remark 16, in order to do that, it is enough to (1) efficiently compute $p_{\boldsymbol{\alpha}}^k(A), p_{\boldsymbol{\alpha}'}^k(A)$ which we already explained how to do, and to (2) efficiently sample from the distribution $q = \frac{(p_{\boldsymbol{\alpha}'}^k - p_{\boldsymbol{\alpha}}^k)_+}{\|p_{\boldsymbol{\alpha}'}^k - p_{\boldsymbol{\alpha}}^k\|}$. We note that $q(A) > 0$ if and only if $N \in A$ in which case

$$
\begin{aligned}
q(A) &= \frac{p_{\boldsymbol{\alpha}'}^k(A) - p_{\boldsymbol{\alpha}}^k(A)}{\|p_{\boldsymbol{\alpha}'}^k - p_{\boldsymbol{\alpha}}^k\|} \\
&= \frac{\pi_{\boldsymbol{\alpha}'}(A)}{\|p_{\boldsymbol{\alpha}'}^k - p_{\boldsymbol{\alpha}}^k\|\Psi_{N,k}^{\boldsymbol{\alpha}'}} - \frac{\pi_{\boldsymbol{\alpha}}(A)}{\|p_{\boldsymbol{\alpha}'}^k - p_{\boldsymbol{\alpha}}^k\|\Psi_{N,k}^{\boldsymbol{\alpha}}} \\
&= \left(\frac{\alpha_N + \delta}{\|p_{\boldsymbol{\alpha}'}^k - p_{\boldsymbol{\alpha}}^k\|\Psi_{N,k}^{\boldsymbol{\alpha}'}} - \frac{\alpha_N}{\|p_{\boldsymbol{\alpha}'}^k - p_{\boldsymbol{\alpha}}^k\|\Psi_{N,k}^{\boldsymbol{\alpha}}} \right) \pi_{\boldsymbol{\alpha}_-}(A \setminus \{N\})
\end{aligned}
$$

Since $\left(\frac{\alpha_N + \delta}{\|p_{\boldsymbol{\alpha}'}^k - p_{\boldsymbol{\alpha}}^k\|\Psi_{N,k}^{\boldsymbol{\alpha}'}} - \frac{\alpha_N}{\|p_{\boldsymbol{\alpha}'}^k - p_{\boldsymbol{\alpha}}^k\|\Psi_{N,k}^{\boldsymbol{\alpha}}} \right)$ does not depend in A, it follows that if $A \sim q$ then $A \setminus \{N\} \sim p_{\boldsymbol{\alpha}_-}^{k-1}$. Hence, in order to sample from q, we can sample a set $\tilde{A} \in \binom{[N-1]}{k-1}$ according to $p_{\boldsymbol{\alpha}_-}^{k-1}$ and then produce the set $A = \tilde{A} \cup \{N\}$

9. Proof of Lemma 21

Proof First, we note that since $D \geq 1$, $D \log\left(\frac{1}{Z}\right) \leq \frac{\tau}{64}$ and $Z \leq \frac{1}{e}$ we have

$$
\tau \geq 64D \log\left(\frac{1}{Z}\right) \geq 64 \geq 8e
$$

Hence, the assumptions in lemmas 18 and 19 are satisfied. Let $G(x) = \int_0^x g(s)ds$. Note that since $0 \leq g(x) \leq 1$ for all x and $g(x) = 0$ for all $x \leq 0$ we have,

$$
G(x) \leq x_+ \tag{20}
$$

Denote $\Phi_t = G(x_t)$ and let $I : \mathbb{R} \to \{0, 1\}$ be the indicator function of the segment $[-2, U + 2]$. By lemma 18 and the assumption that $D \log\left(\frac{1}{Z}\right) \leq \frac{\tau}{64}$ we have

$$
U + 2 \leq \sqrt{16\tau \log\left(\frac{1}{Z}\right)} + 2 \leq \sqrt{16\tau \frac{\tau}{64D}} + 2 \leq \frac{\tau}{2\sqrt{D}} + 2 \leq \frac{\tau}{\sqrt{D}}
$$

In particular, $|x_1| \leq \frac{\tau}{\sqrt{D}}$. Now, by induction we have $|x_t| \leq \frac{\tau}{\sqrt{D}}$ for every t. Indeed, if $|x_t| \leq \frac{\tau}{\sqrt{D}}$ then

$$
|x_{t+1}| = \left| \left(1 - \frac{1}{\tau}\right) x_t + b_t \right| \leq \left(1 - \frac{1}{\tau}\right) \frac{\tau}{\sqrt{D}} + \frac{1}{\sqrt{D}} = \frac{\tau}{\sqrt{D}} .
$$

It follows that $\left|-\frac{x_t}{\tau} + b_t\right| \leq \frac{2}{\sqrt{D}}$. Now, we have

$$
\begin{aligned}
\Phi_{t+1} - \Phi_t &= \int_{x_t}^{x_t - \frac{1}{\tau}x_t + b_t} g(s)ds \\
&\leq g(x_t)\left(-\frac{1}{\tau}x_t + b_t\right) + \frac{1}{2}\left(-\frac{1}{\tau}x_t + b_t\right)^2 \max_{s\in[x_t, x_t - \tau^{-1}x_t + b_t]} |g'(s)| \\
&\leq g(x_t)\left(-\frac{1}{\tau}x_t + b_t\right) + \frac{1}{2}\frac{4}{D}\max_{s\in[x_t, x_t - \tau^{-1}x_t + b_t]} |g'(s)| \\
&\leq g(x_t)\left(-\frac{1}{\tau}x_t + b_t\right) + 2\max_{s\in[x_t, x_t - \tau^{-1}x_t + b_t]} |g'(s)| \\
&= g(x_t)\left(-\frac{1}{\tau}x_t + b_t\right) - 2\max_{s\in[x_t, x_t - \tau^{-1}x_t + b_t]} |g'(s)| + 4\max_{s\in[x_t, x_t - \tau^{-1}x_t + b_t]} |g'(s)| \\
&\leq g(x_t)\left(-\frac{1}{\tau}x_t + b_t\right) - 2\frac{1}{\left|-\frac{1}{\tau}x_t + b_t\right|}|g(x_t) - g(x_{t+1})| + 4\max_{s\in[x_t, x_t - \tau^{-1}x_t + b_t]} |g'(s)| \\
&\leq g(x_t)\left(-\frac{1}{\tau}x_t + b_t\right) - \sqrt{D}|g(x_t) - g(x_{t+1})| + 4\max_{s\in[x_t, x_t - \tau^{-1}x_t + b_t]} |g'(s)| \\
&\leq g(x_t)\left(-\frac{1}{\tau}x_t + b_t\right) - \sqrt{D}|g(x_t) - g(x_{t+1})| + \frac{1}{\tau}x_t g(x_t) I(x_t) + Z
\end{aligned}
$$

Here, the first inequality follows from the fact that for every piece-wise differential function $f : [a,b] \rightarrow \mathbb{R}$ we have $\int_a^b f(x)dx \leq f(a) + \frac{1}{2}(b-a)^2 \max_{\xi\in[a,b]} |f'(\xi)|$. The last inequality follows form lemma 19. Summing from $t=1$ to $t=T$ we get

$$
\Phi_{T+1} - \Phi_1 \leq \sum_{t=1}^T g(x_t)b_t - \sqrt{D}|g(x_t) - g(x_{t+1})| + \sum_{t=1}^T \frac{1}{\tau}x_t g(x_t)(I(x_t) - 1) + TZ
$$

and after rearranging,

$$
-\sum_{t=1}^T g(x_t)b_t + \sqrt{D}|g(x_t) - g(x_{t+1})| \leq \Phi_1 - \Phi_{T+1} + \sum_{t=1}^T \frac{1}{\tau}x_t g(x_t)(I(x_t) - 1) + TZ
$$

Hence, if we denote the loss of the algorithm by L_T, and by L_T^i the loss of the expert i, we have

$$
\begin{aligned}
L_T &= \sum_{t=1}^T g(x_t)l_t(1) + \sum_{t=1}^T (1 - g(x_t))l_t(0) + D|g(x_t) - g(x_{t+1})| \\
&= L_T^0 + \sqrt{D}\left(-\sum_{t=1}^T g(x_t)b_t + \sqrt{D}|g(x_t) - g(x_{t+1})|\right) \\
&\leq L_T^0 + \sqrt{D}\Phi_1 - \sqrt{D}\Phi_{T+1} + \sqrt{D}\sum_{t=1}^T \frac{1}{\tau}x_t g(x_t)(I(x_t) - 1) + \sqrt{D}TZ \qquad (21)
\end{aligned}
$$

In particular, since $\Phi_{T+1} \geq 0$ and $x_t g(x_t)(I(x_t) - 1) \leq 0$ we have

$$
\begin{aligned}
L_T &\leq L_T^0 + \sqrt{D}\Phi_1 + \sqrt{D}TZ \\
&= L_T^0 + \sqrt{D}G(x_1) + \sqrt{D}TZ \\
&\leq L_T^0 + \sqrt{D}x_1 + \sqrt{D}TZ \\
&\leq L_T^0 + \sqrt{D}U + \sqrt{D}2 + \sqrt{D}TZ \\
&\leq L_T^0 + \sqrt{D16\tau \log\left(\frac{1}{Z}\right)} + \sqrt{D}2 + \sqrt{D}TZ
\end{aligned}
$$

This proves the first and last parts of the lemma (as $G(x_1) = 0$ when $x_1 = 0$). It remains to bound $L_T - L_T^1$ when $T \leq \tau$. It is not hard to see that the worst case regret grows as the number of rounds (T) grows but the other parameters remains the same. Hence, we can assume w.l.o.g. that $T = \tau$. Now, note that

$$
\Phi_{T+1} = G(x_{T+1}) = \int_0^{x_{T+1}} g(s)ds \geq x_{T+1} - U
$$

Also,

$$
x_t g(x_t)(1 - I(x_t)) \geq x_t - U - 2 .
$$

Indeed, if $x_t < U$ than the r.h.s. is negative, while the l.h.s. is always non-negative. If $x_t \geq U$ then the l.h.s. equals x_t in which case the inequality is clear. Therefore, if we denote $b_0 := x_1$, we have

$$
\begin{aligned}
\Phi_{T+1} + \sum_{t=1}^{T} \frac{1}{T} x_t g(x_t)(1 - I(x_t)) &\geq (x_{T+1} - U) + \sum_{t=1}^{T} \frac{1}{T}(x_t - U - 2) \\
&= \sum_{j=0}^{T}\left(1 - \frac{1}{T}\right)^{T-j} b_j + \sum_{t=1}^{T} \frac{1}{T} \sum_{j=0}^{t-1}\left(1 - \frac{1}{T}\right)^{t-1-j} b_j - 2(U+1) \\
&= \sum_{j=0}^{T}\left(1 - \frac{1}{T}\right)^{T-j} b_j + \frac{1}{T} \sum_{j=0}^{T-1} \sum_{t=j+1}^{T}\left(1 - \frac{1}{T}\right)^{t-1-j} b_j - 2(U+1) \\
&= \sum_{j=0}^{T}\left(1 - \frac{1}{T}\right)^{T-j} b_j + \frac{1}{T} \sum_{j=0}^{T} \frac{1 - \left(1 - \frac{1}{T}\right)^{T-j}}{\frac{1}{T}} b_j - 2(U+1) \\
&= \sum_{j=0}^{T} b_j - 2(U+1) \\
&= \frac{L_T^0 - L_T^1}{\sqrt{D}} + x_1 - 2(U+1)
\end{aligned}
$$

By equation (21) it follows that

$$
\begin{aligned}
L_T &\leq L_T^1 + \sqrt{D}G(x_1) - \sqrt{D}x_1 + \sqrt{D}2U + \sqrt{D}TZ + 2\sqrt{D} \\
&\leq L_T^1 + \sqrt{D}((x_1)_+ - x_1) + \sqrt{D}2U + \sqrt{D}TZ + 2\sqrt{D} \\
&= L_T^1 + \sqrt{D}(-x_1)_+ + \sqrt{D}2U + \sqrt{D}TZ + 2\sqrt{D} \\
&\leq L_T^1 + \sqrt{D}4 + \sqrt{64DT \log\left(\frac{1}{Z}\right)} + \sqrt{D}TZ
\end{aligned}
$$

$\square$

References

Jacob Abernethy, Peter L Bartlett, Niv Buchbinder, and Isabelle Stanton. A regularization approach to metrical task systems. In *International Conference on Algorithmic Learning Theory*, pages 270–284. Springer, 2010.

Dmitry Adamskiy, Wouter M Koolen, Alexey Chernov, and Vladimir Vovk. A closer look at adaptive regret. In *International Conference on Algorithmic Learning Theory*, pages 290–304. Springer, 2012.

Jason Altschuler and Kunal Talwar. Online learning over a finite action set with limited switching. In *COLT*, 2018.

Lachlan LH Andrew, Siddharth Barman, Katrina Ligett, Minghong Lin, Adam Meyerson, Alan Roytman, and Adam Wierman. A tale of two metrics: Simultaneous bounds on competitiveness and regret. In *COLT*, 2013.

Nikhil Bansal, Niv Buchbinder, Aleksander Madry, and Joseph Naor. A polylogarithmic-competitive algorithm for the k-server problem. *Journal of the ACM (JACM)*, 62(5), 2015.

Yair Bartal, Avrim Blum, Carl Burch, and Andrew Tomkins. A polylog (n)-competitive algorithm for metrical task systems. In *STOC*, 1997.

Yair Bartal, Nathan Linial, Manor Mendel, and Assaf Naor. On metric ramsey-type phenomena. *Annals of Mathematics*, 162:643–709, 2005.

Yair Bartal, Béla Bollobás, and Manor Mendel. Ramsey-type theorems for metric spaces with applications to online problems. *Journal of Computer and System Sciences*, 72(5):890–921, 2006.

A. Blum and Y. Mansour. From external to internal regret. *Journal of Machine Learning Research*, 8:1307–1324, Jun 2007.

Avrim Blum and Carl Burch. On-line learning and the metrical task system problem. *Machine Learning*, 39(1):35–58, 2000.

Avrim Blum, Shuchi Chawla, and Adam Kalai. Static optimality and dynamic search-optimality in lists and trees. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1–8. Society for Industrial and Applied Mathematics, 2002.

A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

Allan Borodin, Nathan Linial, and Michael E Saks. An optimal on-line algorithm for metrical task system. *Journal of the ACM (JACM)*, 39(4):745–763, 1992.

Olivier Bousquet and Manfred K Warmuth. Tracking a small set of experts by mixing past posteriors. *The Journal of Machine Learning Research*, 3:363–396, 2003.

Sébastien Bubeck, Michael B. Cohen, James R. Lee, Yin Tat Lee, and Aleksander Madry. k-server via multiscale entropic regularization. *CoRR*, abs/1711.01085, 2017.

Niv Buchbinder, Shahar Chen, Joseph Seffi Naor, and Ohad Shamir. Unified algorithms for online learning and competitive analysis. In *COLT*, 2012.

Jarosław Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median, and positive correlation in budgeted optimization. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 737–756. SIAM, 2015.

N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

Nicolò Cesa-Bianchi, Yoav Freund, David Haussler, David P. Helmbold, Robert E. Schapire, and Manfred K. Warmuth. How to use expert advice. *Journal of the Association for Computing Machinery*, 44(3):427–485, May 1997.

Nicolo Cesa-Bianchi, Pierre Gaillard, Gábor Lugosi, and Gilles Stoltz. A new look at shifting regret. *CoRR, abs/1202.3323*, 2012.

Nicolo Cesa-Bianchi, Ofer Dekel, and Ohad Shamir. Online learning with switching costs and other adaptive adversaries. In *Advances in Neural Information Processing Systems*, pages 1160–1168, 2013.

Edward G. Coffman and George S. Lueker. *Probabilistic analysis of packing and partitioning algorithms*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, New York, 1991. ISBN 0-471-53272-X. URL http://opac.inria.fr/record=b1088517. A Wiley-Interscience publication.

Amit Daniely, Alon Gonen, and Shai Shalev-Shwartz. Strongly adaptive online learning. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1405–1411, 2015.

Ofer Dekel, Jian Ding, Tomer Koren, and Yuval Peres. Bandits with switching costs: T 2/3 regret. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 459–467. ACM, 2014.

Amos Fiat and Manor Mendel. Better algorithms for unfair metrical task systems and applications. *SIAM Journal on Computing*, 32(6):1403–1422, 2003.

Amos Fiat, Richard M Karp, Michael Luby, Lyle A McGeoch, Daniel D Sleator, and Neal E Young. Competitive paging algorithms. *Journal of Algorithms*, 12(4):685–699, 1991.

Sascha Geulen, Berthold Vöcking, and Melanie Winkler. Regret minimization for online buffering problems using the weighted majority algorithm. In *COLT*, pages 132–143, 2010.

Eric C Hall and Rebecca M Willett. Online optimization in dynamic environments. *arXiv preprint arXiv:1307.5944*, 2013.

Elad Hazan and Tomer Koren. The computational power of optimization in online learning. In *STOC*, 2016.

Elad Hazan and C Seshadhri. Adaptive algorithms for online decision problems. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 14, 2007.

Mark Herbster and Manfred K Warmuth. Tracking the best expert. *Machine Learning*, 32(2):151–178, 1998.

Ali Jadbabaie, Alexander Rakhlin, Shahin Shahrampour, and Karthik Sridharan. Online optimization: Competing with dynamic comparators. *arXiv preprint arXiv:1501.06225*, 2015.

Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 731–740. ACM, 2002.

A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

Michael Kapralov and Rina Panigrahy. Prediction strategies without loss. In *Advances in Neural Information Processing Systems*, pages 828–836, 2011.

Elias Koutsoupias and Christos H Papadimitriou. On the k-server conjecture. *Journal of the ACM (JACM)*, 42(5):971–983, 1995.

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

Haipeng Luo and Robert E Schapire. Achieving all with no parameters: Adaptive normalhedge. In *COLT*, 2015.

Rina Panigrahy and Preyas Popat. Optimal amortized regret in every interval. *arXiv preprint arXiv:1304.7577*, 2013.

Sasha Rakhlin and Karthik Sridharan. Optimization, learning, and games with predictable sequences. In *Advances in Neural Information Processing Systems*, pages 3066–3074, 2013.

Daniel D Sleator and Robert E Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, 2003.