On-line Learning and the Metrical Task System Problem

Avrim Blum[*]
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3891
avrim+@cs.cmu.edu

Carl Burch[†]
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3891
cburch+@cmu.edu

Abstract

We relate two problems that have been explored in two distinct communities. The first is the problem of combining expert advice, studied extensively in the computational learning theory literature, and in particular the problem of tracking the best expert in the clean "decision-theoretic" setting. The second is the Metrical Task System (MTS) problem, studied extensively in the On-line Algorithms literature, and in particular, variations on the setting of the uniform metric space. We show that these problems contain several interesting similarities and demonstrate how algorithms designed for each can be used to achieve good bounds and new approaches for solving the other.

Specific contributions of this paper include:

- An analysis showing how two recent algorithms for the MTS problem can be applied to the setting of tracking the best expert, providing good bounds with an approach of a much different flavor than the well-known multiplicative weighted-expert algorithms.
- A version of Herbster and Warmuth's weight-sharing algorithm for tracking the best expert that works in the decision-theoretic setting, and an application of this to the Metrical Task System problem.
- A new simpler algorithm for tracking experts, which does not carry over to the MTS problem.

Finally, we present an experimental comparison of how these algorithms perform on a process migration problem.

[*]Supported in part by NSF National Young Investigator grant CCR-9357793.

[†]Supported in part by a National Science Foundation Graduate Fellowship.

1 Introduction

Consider a process running on some machine in a network that has the ability to *migrate*: if it finds that its machine is heavily loaded compared to some other machine in the network, it may decide to move to that machine in order to work faster. Let us say that each minute the process quickly pings all n machines in its network to determine their load averages during that minute, and then uses this information to decide whether and where to move for the next minute.

We could imagine modeling the question of when and where such a process should move in the "decision-theoretic" experts framework of Freund and Schapire [FS95] and Chung [Chu94] as follows. We view the loads on the machine as *losses*, where an unloaded machine has loss 0, indicating that the process would have wasted 0 time if it had been on that machine, and a heavily loaded machine has loss approaching 1, indicating the process would have wasted nearly the entire minute had it been on that machine. (Let us assume that the process factors out its influence on the load of its current machine.) At each time step t, the process probabilistically selects a machine to use using some distribution $\mathbf{p}^t$ and then experiences an expected loss $\mathbf{p}^t \cdot \boldsymbol{\ell}^t$, where $\boldsymbol{\ell}^t$ is the vector of losses. Over time, the total expected loss (time wasted) is $\sum_t \mathbf{p}^t \cdot \boldsymbol{\ell}^t$, which we would like to minimize.

One problem with the above model is that moving between machines is not free: it takes time to transfer one's state. Thus, a better model for our problem would be a version of the experts setting in which not only does an algorithm incur loss by selecting a bad expert, but it also incurs loss for moving between experts.

This notion of an on-line algorithm having *state*, with a cost for moving between states, is captured by a problem studied in the On-line Algorithms literature called the *Metrical Task System* (MTS) problem [BLS92, IS95]. In this problem, we imagine the on-line algorithm is controlling a system that can be in one of n *states* or *configurations*. This system is given a sequence of *tasks*, where each task has a *cost vector* specifying the cost of performing the task in each state of the system. There is also a distance metric d among the states of the system specifying the cost of moving from one configuration to another. Given a new task, an algorithm has to decide whether to process the task in the current state (paying the amount specified in the cost vector) or to move to a new state and process the task there (paying both the movement cost and the amount specified in the cost vector

for the new state). Note that in comparison to the experts setting, the MTS algorithm has one-step lookahead, because given a task it moves *first* and then pays (as opposed to the other way around).

One feature of the MTS setting is that, because there is a cost incurred for movement, it makes sense to compare one's performance not just to the best single state (expert), but to the best off-line strategy that is allowed to move between states arbitrarily. That is, one can compare the performance of one's algorithm to the best that one could have done in hindsight if one had known the entire sequence of cost vectors in advance. This measure, and in particular the worst-case of the ratio of the algorithm's performance to the optimal off-line performance, is called the *competitive ratio* of the algorithm.

This paper begins in Section 2 by more precisely defining the problems and goals and describing some general relationships. In Section 3 we present several algorithms designed for the MTS problem and show how these can be applied to the problem of tracking the best expert in the decision-theoretic setting. In the other direction, in Section 4 we present two algorithms for tracking the best expert in the decision-theoretic setting based on established weighted-experts algorithms [HW95, LW94] and show that one of them achieves good performance for the MTS problem, whereas the other has an unbounded competitive ratio. Finally, in Section 5 we present an empirical comparison of these algorithms and others for the process migration problem.

2 Definitions and general relations

2.1 The MTS problem and the competitive ratio

In the *Metrical Task System (MTS) problem* [BLS92, IS95], an on-line algorithm controls a system with n *states* located at points in a space with distance metric d. The algorithm receives, one at a time, a sequence of *tasks*, each a *cost vector* specifying the cost of performing the task in each state. Say the system currently occupies state i and, given the task vector $\boldsymbol{\ell}$, the algorithm tells the system to move to state j. Then the algorithm pays both the cost of moving to state j and the amount specified by the cost vector for processing the task in j; that is, the cost is $d_{i,j} + \ell_j$. (If $i = j$ then, since $d_{i,i} = 0$, the cost is just ℓ_j.)

In this paper we consider the simplest case of the MTS problem, in which we have a uniform metric space. In other words, $d_{i,j}$ is 0 if $i = j$ and 1 otherwise.

A feature of the MTS problem is that, because of the cost for movement, one can feasibly compare the performance of an on-line algorithm to the best that one could have done in hindsight if one had known the task sequence in advance (this is called the optimal off-line cost). The *competitive ratio* is the worst-case ratio of the on-line to the optimal off-line performance. Specifically, say $c_A(\sigma)$ is the cost to algorithm A for task sequence σ; if A is randomized, then for a fixed σ, $c_A(\sigma)$ is a random variable. We say that algorithm A has competitive ratio a if, for some b, for all task sequences σ,

$$\mathbf{E}[c_A(\sigma)] \leq a \cdot c_{OPT}(\sigma) + b\,,$$

where OPT is the optimal off-line algorithm (the optimal strategy in hindsight).

This definition is often termed the *oblivious adversary* model since the order of quantifiers ($\forall\sigma \mathbf{E}[c_A(\sigma)]$) can be thought of as an adversary that must choose σ in advance, before seeing the outcome of A's coin flips. A consequence of this model (as opposed to one in which the adversary may choose tasks based on the algorithm's previous coin flips) is that in the uniform metric space, to move from probability distribution $\mathbf{p}^t$ to distribution $\mathbf{p}^{t+1}$ the algorithm need pay only

$$\sum_i \max\{0, p_i^t - p_i^{t+1}\}$$

(rather than the cost that would be incurred by choosing its position at time $t+1$ completely independently of its position at time t). In other words, say at time t the randomized on-line algorithm has a probability distribution $\mathbf{p}^t$, receives a cost vector $\boldsymbol{\ell}^t$, and modifies its distribution to $\mathbf{p}^{t+1}$. Then, its total cost is

$$\sum_i \max\{0, p_i^t - p_i^{t+1}\} + \mathbf{p}^{t+1} \cdot \boldsymbol{\ell}^t\,.$$

For a refined analysis of the MTS problem, we use the *r-unfair competitive ratio* considered in [BKRS92] and formalized explicitly in [Sei96]. Here the on-line algorithm pays the same amount as before, but OPT pays r times more for movement. That is, the off-line player pays $rd_{i,j} + \ell_j$ for a task. This parameter r is known to the on-line and off-line MTS algorithms.

2.2 Tracking experts in the decision-theoretic setting

The second setting we consider is the "decision-theoretic" framework for learning from expert advice (also called the on-line allocation problem) [FS95, Chu94]. In this problem the learning algorithm faces a sequence of *trials*. For trial t, the algorithm chooses a probability distribution $\mathbf{p}^t$ over a set of n *experts*. After choosing this distribution, a *loss vector* $\boldsymbol{\ell}^t \in [0,1]^n$ is revealed, and the learning algorithm is penalized with loss $\mathbf{p}^t \cdot \boldsymbol{\ell}^t$. This penalty may be thought of as the expected loss of the algorithm if it selects one expert with distribution $\mathbf{p}^t$ and then incurs that expert's loss. We will call this the Experts-DTF setting.

This problem is typically analyzed with the goal of performing nearly as well as the best expert on the given sequence of trials. We consider the *partitioning bound*, a stronger goal (considered by [HW95] and [LW94] for specific classes of loss functions) of performing nearly as well as the best sequence of experts. Specifically, given a partition of the trial sequence into k segments, we define the *loss of the partition* as $\sum_{i=1}^{k} L[i]$, where $L[i]$ is the loss of the best expert in segment i. Our goal is to achieve, for every partition P, a loss at most $aL_P + bk_P$, where L_P is the loss of the partition and k_P is the number of segments it has. We will seek to minimize a and b.

Notice that the MTS problem is much like the Experts-DTF problem, but with a few key differences:

- The MTS problem contains a cost for switching between states/experts.
- The MTS problem has *one-step lookahead*. That is, first the cost vector is announced, then the algorithm

chooses whether to move, and finally the algorithm pays according to the entry in the cost vector for the new state. In contrast, the Experts-DTF algorithm has *zero lookahead*, in that it first pays and then moves.

- Because of the lookahead, MTS algorithms can deal with unbounded cost vectors. Large losses are actually advantageous to an on-line MTS algorithm in the sense that an adversary can hide this information from the algorithm by splitting the task into many smaller pieces.

2.3 A general relation

The problem of achieving a good r-unfair competitive ratio for the uniform MTS problem is closely related to the partitioning bound in the Experts-DTF setting, as evidenced by the following theorem.

Theorem 1 *Let A be a randomized algorithm that, given r, achieves competitive ratio $a(n,r)$ for the r-unfair uniform MTS problem. Then in the Experts-DTF setting, for any given r, A can achieve total loss at most*

$$a(n,r)L + r \cdot a(n,r)k + b,$$

for any k-segment partition of loss L, for some constant b that may depend on r and n (typically, $b \leq r$).

Proof. Consider some sequence of trials σ. Let L_A denote the loss of algorithm A on σ and let c_A denote the expected cost of A on σ viewed as tasks in a uniform MTS problem. Then, $L_A \leq c_A$ for the following reason. Consider a single trial. Let ℓ_i be the loss of expert i, and let p_i and p'_i represent the probability assigned to expert i before and after the trial. Then the loss in the experts setting can be bounded as follows:

$$\begin{aligned} L_A &= \sum_i p_i \ell_i \\ &= \sum_i p'_i \ell_i + \sum_i (p_i - p'_i)\ell_i \\ &\leq \sum_i p'_i \ell_i + \sum_{i:p_i > p'_i} (p_i - p'_i)\ell_i \\ &\leq \sum_i p'_i \ell_i + \sum_{i:p_i > p'_i} (p_i - p'_i) \\ &= c_A. \end{aligned}$$

Now consider an off-line strategy that switches between experts k times and incurs loss L. Its cost in the r-unfair MTS setting is $L + rk$. Therefore, $L_A \leq a(n,r)(L + rk) + b$ (for some constant b) as desired. ∎

In particular, an immediate consequence of Theorem 1 is that if $a(n,r)$ is of the form $(1 + (\alpha \ln n)/r)$, then for any given $\epsilon > 0$, by setting $r = (\alpha \ln n)/\epsilon$, A can achieve total loss at most

$$(1+\epsilon)L + \left(\frac{(1+\epsilon)\alpha \ln n}{\epsilon}\right) k + b$$

(where typically $b \leq \frac{\alpha \ln n}{\epsilon}$).

In the other direction, algorithms that achieve good bounds in the Experts-DTF setting may or may not perform well on the MTS problem, depending on how well they control their movement costs. We will see that one such algorithm, a weight-sharing algorithm based on the ideas of Herbster and Warmuth succeeds in the MTS setting, whereas another algorithm based on Littlestone and Warmuth's WML strategy does not.

3 MTS-style algorithms

We now describe several algorithms designed for the MTS problem, and show how they can be applied to the Experts-DTF setting.

To describe these algorithms, we need to define the notion of the *reduced loss* of an expert (usually called the *work function* in the On-line Algorithms literature). For a parameter r, the reduced loss $\hat{L}_i^t$ of expert i at time t is defined as follows. Initially $\hat{L}_i^0$ is 0. Given a loss vector $\boldsymbol{\ell}^t$ we update each expert's reduced loss to

$$\hat{L}_i^{t+1} = \min\{\hat{L}_i^t + \ell_i^t, \min_{j \neq i} \hat{L}_j^t + \ell_j^t + r\}.$$

(Using MTS terminology, $\hat{L}_i^t$ is the optimal off-line cost for servicing the first t requests and ending at state i, in the r-unfair setting where the off-line algorithm pays r to switch its state.) Notice that $\hat{L}_i$ is never greater than the actual total loss incurred by expert i, and in particular $\hat{L}_i$ and $\hat{L}_j$ can never differ by more than r. If $\hat{L}_i = \hat{L}_j + r$, then so long as expert j does not incur a loss, expert i can incur losses with impunity without increasing its reduced loss. To emphasize this point, if one reduced loss is r greater than the other, we will say that the higher is *pinned* by the lower.

3.1 Two experts

The following algorithm of [BKRS92] achieves an optimal competitive ratio for the r-unfair MTS problem on two states. In the Experts-DTF setting, the algorithm can be viewed as follows:

Algorithm Linear **[BKRS92]:** *The algorithm has one parameter r. Let $\hat{L}_i$ represent the reduced loss of expert i with respect to r. The algorithm allocates*

$$p_0 = \frac{1}{2} + \frac{\hat{L}_1 - \hat{L}_0}{2r}$$

probability to expert 0 and the rest to expert 1 (whose probability equation is symmetric).

This strategy moves probability linearly between experts, so that an expert's probability is zero when it is pinned. This strategy's MTS performance problem turns out to be optimal and is summarized by the following theorem.

Theorem 2 ([BKRS92]) Linear *has r-unfair competitive ratio $1 + 1/r$ on the two-state MTS problem.*

A corollary of this, using Theorem 1, is that the partitioning bound of Linear for two experts is at most

$$(1 + 1/r)L + (r+1)k + b$$

(where in fact $b = r/2$). For a somewhat better bound we analyze the algorithm directly in the Experts-DTF setting.

Theorem 3 *For integer parameter r, the loss of* Linear *for two experts is at most*

$$\left(1+\frac{1}{2r}\right)L+\left(r+\frac{1}{2}\right)k$$

for any k-segment partition of loss L.

Proof. Consider segment i of the partition with a loss $L[i]$. Assume w.l.o.g. that that the better expert for the segment is expert 0. (So $L[i]$ represents the total loss to expert 0 in the segment.) Let δ represent the fractional component of $|\hat{L}_1-\hat{L}_0|$. That is, $\delta=(\hat{L}_1-\hat{L}_0)-\lfloor\hat{L}_1-\hat{L}_0\rfloor$. We will use a potential function over this segment of[1]

$$\Phi=rp_1^2+\frac{1}{2}p_1+\frac{\delta(1-\delta)}{4r}\,.$$

Notice that Φ is always between 0 and $r+1/2$.

Say the algorithm receives loss vector $\langle\ell_0,\ell_1\rangle$. Our goal is to show that the algorithm's cost plus potential change is at most $\ell_0(1+1/2r)$. The easiest case is when either $\hat{L}_0-\hat{L}_1=r$ and $\ell_0\geq\ell_1$ or $\hat{L}_0-\hat{L}_1=-r$ and $\ell_0\leq\ell_1$; either way, Φ does not change and the algorithm's cost is $\min(\ell_0,\ell_1)\leq\ell_0<\ell_0(1+1/2r)$. We split the remaining possibilities into 4 cases:

Case 1: Suppose $\ell_0\geq\ell_1$ and $\hat{L}_0-\hat{L}_1\leq r-\ell$, where $\ell=\ell_0-\ell_1$. Then p_1 gains $\ell/2r$ probability. Notice that the last term of the potential function increases most when δ is initially 0. So the amortized cost for the trial is

$$\begin{aligned}p_0\ell_0+p_1\ell_1+\Delta\Phi &\leq \ell_1+p_0\ell+\left(p_1\ell+\tfrac{\ell^2}{4r}+\tfrac{\ell}{4r}+\tfrac{\ell(1-\ell)}{4r}\right)\\ &= \ell_1+\ell\left(1+\tfrac{1}{2r}\right)\\ &\leq \ell_0\left(1+\tfrac{1}{2r}\right).\end{aligned}$$

Case 2: Suppose $\ell_0\geq\ell_1$ but unlike Case 1, $\hat{L}_0-\hat{L}_1=r-\tilde{\ell}>r-\ell$. Then p_1 increases from $1-\tilde{\ell}/2r$ to 1, and δ drops from $\tilde{\ell}$ to 0. The amortized cost for the trial is

$$\begin{aligned}p_0\ell_0+p_1\ell_1+\Delta\Phi &= \ell_1+\tfrac{\tilde{\ell}}{2r}\ell+\left(\tilde{\ell}-\tfrac{\tilde{\ell}^2}{4r}+\tfrac{\tilde{\ell}}{4r}-\tfrac{\tilde{\ell}(1-\tilde{\ell})}{4r}\right)\\ &\leq \ell_1+\ell\left(1+\tfrac{1}{2r}\right)\\ &\leq \ell_0\left(1+\tfrac{1}{2r}\right).\end{aligned}$$

Case 3: As the flip side of Case 1, suppose $\ell_0<\ell_1$ and $\hat{L}_1-\hat{L}_0\leq r-\ell$, where $\ell=\ell_1-\ell_0$. Then p_1 loses $\ell/2r$ probability. The last term of the potential function increases by at most $\ell(1-\ell)/4r$. So the cost is:

$$\begin{aligned}p_0\ell_0+p_1\ell_1+\Delta\Phi &\leq \ell_0+p_1\ell+\\ &\qquad\left(-p_1\ell+\tfrac{\ell^2}{4r}-\tfrac{\ell}{4r}+\tfrac{\ell(1-\ell)}{4r}\right)\\ &= \ell_0\\ &< \ell_0\left(1+\tfrac{1}{2r}\right).\end{aligned}$$

Case 4: Suppose $\ell_0<\ell_1$ but unlike Case 3, $\hat{L}_1-\hat{L}_0=r-\tilde{\ell}>r-\ell$. Then p_1 drops from $\tilde{\ell}/2r$ to 0, and, because r is integral, δ drops from $1-\tilde{\ell}$ to 0. The amortized cost for the trial is

$$\begin{aligned}p_0\ell_0+p_1\ell_1+\Delta\Phi &= \ell_0+\tfrac{\tilde{\ell}}{2r}\ell+\left(-\tfrac{\tilde{\ell}^2}{4r}-\tfrac{\tilde{\ell}}{4r}-\tfrac{(1-\tilde{\ell})\tilde{\ell}}{4r}\right)\\ &\leq \ell_0\\ &< \ell_0\left(1+\tfrac{1}{2r}\right).\end{aligned}$$

In all cases, the algorithm's cost is at most $\ell_0(1+1/2r)$. So the total cost for the ith segment is $(1+1/2r)L[i]$ plus the maximum potential change between segments, $r+1/2$. Thus the total cost for the partition is

$$\sum_{i=1}^{k}\left(\left(1+\frac{1}{2r}\right)L[i]+r+\frac{1}{2}\right)=\left(1+\frac{1}{2r}\right)L+\left(r+\frac{1}{2}\right)k$$

as desired. ∎

[1] If the losses are always in $\{0,1\}$ then the proof can be simplified by ignoring δ (it will always be 0) and ignoring cases 2 and 4 (which will never occur).

3.2 Marking

For the MTS problem on a uniform metric space of more than two states, the standard algorithm is the Marking algorithm of Borodin, Linial, and Saks [BLS92] and Fiat et al. [FKL+91].

Algorithm Marking **[BLS92]:** *We maintain a counter for each state. At the beginning of each phase, the counters are reset to 0, and the algorithm occupies a random state. Given a cost vector $\boldsymbol{\ell}$, we increment the ith counter by ℓ_i. If the counter for our state reaches r, we move to a random state whose counter is less than r. If no such state exists, we begin a new phase by resetting the counters and going to a random state.*

This algorithm was designed for the "fair" setting in which $r=1$. For that case, its competitive ratio is $2H_n$ (where $H_n\in[\ln n,\ln n+1]$ is the nth harmonic number), which is optimal to constant factors [BLS92]. For the "unfair" setting, however, the competitive ratio does not decrease substantially with r as we would like: the ratio becomes $(1+1/r)H_n$. Therefore, the partitioning bound resulting from Theorem 1 is not so good.

Corollary 4 *The loss of* Marking *for n experts will be at most*

$$H_n(1+1/r)L+H_n(r+1)k+r$$

for any k-segment partition of loss L.

In fact, it is easy to construct examples where the loss of Marking is $\Omega(L\log n)$.

In contrast, we describe below an algorithm whose competitive ratio is $1+O(\log(n)/r)$, resulting in a much better bound for the Experts-DTF problem.

3.3 Odd-Exponent

The good performance of Linear suggests using a generalization for more than two experts. Bartal, Blum, Burch, and Tomkins [BBBT97] analyze a generalization that can also be applied in an experts setting:

Algorithm Odd-Exponent **[BBBT97]:** *Let t be an odd integer, and let $\hat{L}_i$ represent the reduced loss of expert i. Then place*

$$p_i=\frac{1}{n}+\frac{1}{n}\sum_{j=1}^{n}\left(\frac{\hat{L}_j-\hat{L}_i}{r}\right)^t$$

probability on expert i.

The following theorem shows that this strategy performs well in a uniform task system.

Theorem 5 ([BBBT97]) *For an n-node uniform task system,* Odd-Exponent *has an r-unfair competitive ratio of $1 + 2n^{1/t}t/r$.*

In particular, notice that, if we let t be the odd integer closest to $\ln n$, then the bound provided by Theorem 5 is approximately $1 + 2e \ln n/r$.

If we strictly use reduced loss in Odd-Exponent's probability distribution, we could allocate negative probability to an expert. (Consider the case where one expert has reduced loss of r while the rest are zero.) The analysis of [BBBT97], concerned with theoretical guarantees, skirts the issue by observing that we may assume without loss of generality that experts with zero probability incur zero loss, and furthermore that because of the one-step lookahead in the MTS setting, that an expert never receives a greater loss than that needed to set its probability to zero.

If we wish to implement the algorithm, however, we must confront the possibility that tasks observed will not obey this condition. We can do this by using a modification of reduced loss, $\tilde{L}$, in computing the probability distribution of the strategy. Say the strategy receives a loss vector $\boldsymbol{\ell}_i$. We will change $\tilde{L}_i$ to become, not $\min\{\tilde{L}_i + \ell_i, \min_j \tilde{L}_j + \ell_j + r\}$ as for reduced loss, but $\min\{\tilde{L}_i + \ell_i, x\}$, where x is the greatest value such that no probabilities are negative. (In an implementation one can compute x by using numerical techniques to find where the function returning the minimum probability for a given x reaches zero.) This avoids negative probabilities because each probability that would have become negative with the unmodified reduced loss becomes zero instead.

This modification maintains the same competitive ratio for the following reason. We can think of this modification as dividing each cost vector into two pieces, $\tilde{\boldsymbol{\ell}}$ and $\boldsymbol{\ell} - \tilde{\boldsymbol{\ell}}$, where $\tilde{\boldsymbol{\ell}} = \tilde{L}^{t+1} - \tilde{L}^t$. For $\tilde{\boldsymbol{\ell}}$, the algorithm will pay at most $1 + 2n^{1/t}t/r$ times the off-line player would on $\tilde{\boldsymbol{\ell}}$ (which itself is less than the off-line player pays on $\boldsymbol{\ell}$). For $\boldsymbol{\ell} - \tilde{\boldsymbol{\ell}}$, the algorithm will pay nothing, since the vector is non-zero only at states where $\tilde{L} = x$, and these states have no probability.

Theorems 1 and 5 immediately imply a bound for the Experts-DTF setting.

Corollary 6 *Let c be a positive real number. Configure* Odd-Exponent *with t the odd integer closest to $\ln n$ and $r = (2n^{1/t}t)c$, and predict according to its probability distribution. Then the partitioning bound for n experts will be at most*

$$\left(1 + \frac{1}{c}\right) L + \left(2(c+1)n^{1/t}t\right) k + 2cn^{1/t}t$$
$$\approx \left(1 + \frac{1}{c}\right) L + \left(2e(c+1)\ln n\right) k + 2ec \ln n,$$

for any k-segment partition of loss L.

Proof. This is a corollary of Theorem 5 by applying Theorem 1. The additive term comes from the fact that in the analysis of [BBBT97], the algorithm competes against not the minimum reduced loss, but the average, which is at most r more. ∎

Unfortunately, for the same L coefficient, the k coefficient guaranteed by Corollary 6 is approximately $2e$ times worse than for an algorithm based on Herbster and Warmuth's weight-sharing algorithm [HW95]. Because the competitive ratio of Theorem 5 is not known to be tight, and because, as demonstrated by the direct analysis of Linear, taking a competitive ratio through Theorem 1 is not necessarily best, the actual performance may be better than these theorems indicate.

4 Multiplicative weight-updating algorithms

As we observed, an analog of Theorem 1 does not hold for Experts-DTF algorithms in the MTS setting, since they may not control their movement costs well. We examine one such algorithm, based on Littlestone and Warmuth's WML, that works well in the Experts-DTF but not the MTS settings. We will see that another algorithm, a weight-sharing algorithm based on the ideas of Herbster and Warmuth, succeeds in both settings.

4.1 A threshold experts algorithm

The WML algorithm, due to Littlestone and Warmuth, is the following strategy for the on-line discrete prediction problem.

Algorithm WML **[LW94]:** *The algorithm uses two parameters, $\beta \in [0, 1]$ and $\alpha \in [0, 1/2]$, and maintains a weight w_i for each expert, initialized to 1. When an expert makes a mistake, we multiply its weight w_i by β, but only if its weight is at least $\alpha W/n$, where $W = \sum w_i$. We then select the prediction on which the most weight falls.*

The obvious generalization to the Experts-DTF setting is to place w_i/W probability on expert i. However, this does not result in a good worst-case bound. Consider a two-expert example where expert 0 incurs no loss and expert 1 has a loss of one for each trial. Then w_0 remains at one while w_1 stops decreasing when it reaches $\beta\alpha/(2 - \beta\alpha)$. For each subsequent trial the algorithm incurs at least $\beta\alpha/2$ loss, so the loss is arbitrarily large.

This motivates a different generalization of the algorithm.

Algorithm Thresh**:** *As in* WML*, this algorithm uses parameters $\beta \in [0, 1]$ and $\alpha \in [0, 1/2]$, and maintains a weight w_i on each expert, initialized to 1. When expert i incurs a loss of ℓ_i, we multiply its weight w_i by β^{ℓ_i} but only if its weight is at least $\alpha W/n$, where $W = \sum w_i$. Let $\hat{W}$ be the sum of weights w_i that are at least $\alpha W/n$. Put zero probability on expert i if $w_i < \alpha W/n$ and put $w_i/\hat{W}$ probability on expert i otherwise.*

4.1.1 Experts-DTF performance

Theorem 7 *Given n experts,* Thresh *will incur a loss of at most*

$$\left(\frac{\ln(1/\beta)}{(1-\beta)(1-\alpha)}\right) L + \left(\frac{\ln(n/\beta\alpha)}{(1-\beta)(1-\alpha)}\right) k$$

for any k-segment partition of loss L.

Proof. Consider a loss vector $\boldsymbol{\ell}^t$ for trial t. Remembering that $\hat{W} \geq W(1-\alpha)$, we can bound how the total weight W^t changes after each trial.

$$\begin{aligned} W^{t+1} &= \sum_{w_i^t \geq \alpha W^t/n} \beta^{\ell_i^t} w_i^t + \sum_{w_i^t < \alpha W^t/n} w_i^t \\ &\leq \sum_{w_i^t \geq \alpha W^t/n} (1-(1-\beta)\ell_i^t) w_i^t + \sum_{w_i^t < \alpha W^t/n} w_i^t \\ &= W^t \left(1 - (1-\beta) \sum_{w_i^t \geq \alpha W^t/n} \frac{w_i^t}{W^t} \ell_i^t\right) \\ &\leq W^t \left(1 - (1-\beta)(1-\alpha) \sum_{w_i^t \geq \alpha W^t/n} \frac{w_i^t}{\hat{W}^t} \ell_i^t\right) \\ &= W^t \left(1 - (1-\beta)(1-\alpha) \mathbf{p}^t \cdot \boldsymbol{\ell}^t\right) \end{aligned}$$

Consider a segment in which expert i incurs a total loss of L_i. Say that the total weight at the beginning of the segment is W^0. Then, since the initial weight of expert i is at least $\beta\alpha W^0/n$, its weight after the segment is at least $\beta^{L_i}\beta\alpha W^0/n$. If we let W^{final} represent the total weight after the segment ends, we can bound our total loss for the segment, $\sum_t \mathbf{p}^t \cdot \boldsymbol{\ell}^t$.

$$\begin{aligned} \frac{\beta^{L_i}\beta\alpha W^0}{n} &\leq W^{final} \\ &\leq W^0 \prod_t \left(1 - (1-\beta)(1-\alpha)\mathbf{p}^t \cdot \boldsymbol{\ell}^t\right). \end{aligned}$$

So,

$$L_i \ln\beta + \ln\frac{\beta\alpha}{n} \leq -(1-\beta)(1-\alpha)\sum_t \mathbf{p}^t \cdot \boldsymbol{\ell}^t$$

and thus

$$\sum_t \mathbf{p}^t \cdot \boldsymbol{\ell}^t \leq \frac{\ln(1/\beta)}{(1-\beta)(1-\alpha)} L_i + \frac{\ln(n/\beta\alpha)}{(1-\beta)(1-\alpha)}.$$

Therefore the total cost to Thresh over the partition is at most

$$\frac{\ln(1/\beta)}{(1-\beta)(1-\alpha)} L + \frac{\ln(n/\beta\alpha)}{(1-\beta)(1-\alpha)} k$$

for a k-segment partition of loss L. ∎

4.1.2 MTS performance

This result, unfortunately, does not carry over to the MTS setting. Consider the two-expert case. Say that expert 1 incurs a large enough loss for its weight to drop to slightly below $\alpha/(2-\alpha)$. At this point, the algorithm has all probability on expert 0. Now, suppose expert 0 incurs a tiny loss, just sufficient to bring w_1 to equal $\alpha W/n$. This forces the algorithm to move $\alpha/2$ probability over to expert 1. Now suppose expert 1 incurs an infinitesimal loss so that $w_1 < \alpha W/n$. This forces the algorithm to move $\alpha W/n$ probability back to expert 0. This situation can repeat indefinitely, causing the algorithm to incur unbounded movement cost with insignificant increase in the off-line optimal cost, giving an unbounded competitive ratio.

4.2 A weight-sharing experts algorithm

We now describe a weight-sharing algorithm based on the Variable-share algorithm of Herbster and Warmuth [HW95] and prove that it achieves good bounds in both the Experts-DTF and MTS settings.

Algorithm Share**:** *The algorithm has two parameters: $\beta \in [0,1]$ is the usual penalty parameter and $\alpha \in [0, 1/2]$ is a "sharing" parameter. The algorithm maintains n weights $w_1, \ldots, w_n$, each initialized to 1. The weights are used to determine the probability distribution in the usual way, namely $p_i = w_i/W$, where $W = \sum_i w_i$. Given a loss vector $\boldsymbol{\ell}$ the algorithm updates the weights using the formula:*

$$w_i \leftarrow w_i\beta^{\ell_i} + \alpha\Delta/n\,,$$

where $\Delta = \sum_i (w_i - w_i\beta^{\ell_i})$.

The update rule used by this algorithm can be viewed as follows. We first update as usual: $w_i \leftarrow w_i\beta^{\ell_i}$. This reduces the sum of the weights by some amount Δ. We then distribute an α fraction of this Δ evenly among the n experts ($\alpha\Delta/n$ each).

4.2.1 Experts-DTF performance

Theorem 8 *Given n experts,* Share *incurs a loss at most*

$$\left(\frac{\ln(1/\beta)}{(1-\beta)(1-\alpha)}\right) L + \left(\frac{\ln((n+2)/\alpha)}{(1-\beta)(1-\alpha)}\right) k\,,$$

for any k-segment partition of loss L.

Proof. Consider a specific segment of the given sequence of trials. Our goal is to show that, for any expert i, the loss L_A incurred by the algorithm in this segment is at most $(\ln(1/\beta)L_i + \ln((n+2)/\alpha))/(1-\beta)(1-\alpha)$, where L_i is the loss incurred by expert i in the segment.

Let us define at time t the algorithm's probability distribution to be $\mathbf{p}^t$, the loss vector to be $\boldsymbol{\ell}^t$, and the sum of the weights to be W^t. Then, using the same analysis as for Thresh, we have:

$$W^{t+1} \leq W^t(1-(1-\beta)(1-\alpha)\mathbf{p}^t \cdot \boldsymbol{\ell}^t).$$

So, if W^0 is the sum of weights at the start of the segment and W^{final} is the sum of weights at the end of the segment, then

$$W^{final} \leq W^0 \prod_t (1-(1-\beta)(1-\alpha)\mathbf{p}^t \cdot \boldsymbol{\ell}^t)\,. \quad (1)$$

Now consider the weight of expert i. In the update made at time t, the amount added to w_i due to the share update is $\frac{\alpha(W^t - W^{t+1})}{(1-\alpha)n}$. (This is because $W^{t+1} = W^t - \Delta^t + \alpha\Delta^t$, where Δ^t is the value of Δ specified in the algorithm at time t.) In the entire segment, therefore, the total amount added to w_i due to the share updates is $\frac{\alpha(W^0 - W^{final})}{(1-\alpha)n}$. Thus, even if w_i is zero at the start of the segment, by the end of the segment we have:

$$w_i^{final} \geq \beta^{L_i}\left(\frac{\alpha(W^0 - W^{final})}{(1-\alpha)n}\right). \quad (2)$$

For convenience, define

$$\Pi = \prod_t (1 - (1-\beta)(1-\alpha)\mathbf{p}^t \cdot \boldsymbol{\ell}^t).$$

Combining equations 1 and 2 we get:

$$\Pi \geq \frac{\beta^{L_i}\alpha(1-\Pi)}{(1-\alpha)n}.$$

We can now solve for Π.

$$\Pi \geq \frac{\beta^{L_i}\alpha/(1-\alpha)n}{1+\beta^{L_i}\alpha/(1-\alpha)n}$$

So,

$$\begin{aligned} -\ln\Pi &\leq \ln\left(\frac{(1-\alpha)n}{\beta^{L_i}\alpha}\right) + \frac{\beta^{L_i}\alpha}{(1-\alpha)n} \\ &\leq \ln\left(\frac{(1-\alpha)n}{\beta^{L_i}\alpha}\right) + \frac{1}{n} \\ &\leq L_i \ln(1/\beta) + \ln\left(\frac{n+2}{\alpha}\right) \end{aligned}$$

Recalling the definition of Π, we notice that

$$-\ln\Pi \geq (1-\beta)(1-\alpha)L_A\,,$$

so

$$L_A \leq \frac{L_i\ln(1-\beta) + \ln((n+2)/\alpha)}{(1-\beta)(1-\alpha)}$$

as desired. ∎

4.3 MTS performance

Theorem 9 *For a given r, configure* Share *with $\alpha = 1/(2r+1)$ and $\beta = \max\{1/2, 1-\gamma\}$, where $\gamma = \ln((n+2)/\alpha)/r$. Then in the uniform MTS setting, the r-unfair competitive ratio of* Share *is at most $(1 + 8\ln((n+2)(2r+1))/r)$.*

(Note: these constants have not been optimized; the formulas used for α and β are designed mainly to simplify the analysis.)

Proof. Suppose that the optimal off-line strategy makes k moves on the given task sequence. Let OPT^t denote the cost incurred by this strategy between its $(t-1)$st and tth moves, and let $\text{OPT}_{local} = \sum_{t=1}^{k+1}\text{OPT}^t$. Then we can write the optimal off-line cost OPT as

$$\text{OPT} = \text{OPT}_{local} + rk.$$

Let us consider separately the local cost $cost_{local}$ and the movement cost $cost_{move}$ of the Share algorithm. We have shown (Theorem 8) that the local cost satisfies:

$$cost_{local} \leq \text{OPT}_{local}\left(\tfrac{\ln(1/\beta)}{(1-\beta)(1-\alpha)}\right) + k\left(\tfrac{\ln((n+2)/\alpha)}{(1-\beta)(1-\alpha)}\right) \quad (3)$$

(In fact, the MTS problem allows one-step lookahead, which only helps the algorithm.) Define $\epsilon = 1-\beta$, so $\epsilon = \min\{1/2, \gamma\}$. If $\epsilon = \gamma$, by equation 3 we have:

$$\begin{aligned} cost_{local} &\leq \text{OPT}_{local}(1+\epsilon/2+\epsilon^2/3+\ldots)/(1-\alpha) \\ &\quad + kr/(1-\alpha) \\ &\leq \text{OPT}_{local}(1+\epsilon)/(1-\alpha) + kr/(1-\alpha) \\ &\leq \text{OPT}(1+\gamma)/(1-\alpha). \end{aligned}$$

If $\epsilon = 1/2$, then, since $\gamma > 1/2$, we have

$$\begin{aligned} cost_{local} &\leq \text{OPT}_{local}(1+\epsilon)/(1-\alpha) + 2kr\gamma/(1-\alpha) \\ &\leq \text{OPT}(1+2\gamma)/(1-\alpha) \end{aligned}$$

Now we would like to show that the movement cost is bounded by $(3\epsilon/2)cost_{local}$. If we can get this, then we can bound our total cost by

$$\begin{aligned} \text{OPT}(1+2\gamma)(1+3\epsilon/2)/(1-\alpha) &< \text{OPT}(1+5\gamma)/(1-\alpha) \\ &< \text{OPT}(1+8\gamma) \end{aligned}$$

which is our goal.

Because the MTS setting has one-step lookahead, one can assume for convenience that each task (loss) vector is *δ-elementary*, meaning that it is zero in all states except one, and in that state, the task has cost δ, where δ can be chosen to be arbitrarily small.

Consider now some δ-elementary task vector, with non-zero cost in state i. Say that before processing this task, the algorithm had weight w_i at point i and the total weight was W, so the probability p_i at point i was w_i/W. The movement cost incurred by the algorithm is $p_i - p_i'$, where p_i' is the probability at point i after the move. We can lower-bound p_i' by

$$p_i' \geq \frac{w_i\beta^\delta}{W}$$

because the true numerator in the formula for p_i' is larger than this (since it includes the share) and the true denominator is smaller (since W decreased). So, $p_i - p_i' \leq p_i(1-\beta^\delta)$ and therefore the total movement cost satisfies:

$$\begin{aligned} cost_{move} &\leq p_i(1-\beta^\delta) \\ &\leq p_i(1-(1-3\delta(1-\beta)/2)) \\ &\leq \tfrac{3}{2}(1-\beta)cost_{local}, \end{aligned}$$

where the last inequality holds because $cost_{local} = p_i\delta$ (actually, $cost_{local} = p_i'\delta$, but our above bounds on $cost_{local}$ hold for the case in which p_i is used instead of p_i'). ∎

5 Process migration simulation

We now describe some brief experimental results comparing the algorithms analyzed above and others on data representing a process migration problem. Process migration has aspects of both the MTS problem and the experts setting. There is a cost to move between machines, but there is also zero lookahead.

As data for process migration, we collected load averages for several (112) machines around the CMU campus. We queried each machine every five minutes for 6.5 days. From these machines, we selected 32 that were busy enough to be interesting for this analysis.

Each five-minute interval corresponds to a trial with loss vector $\boldsymbol{\ell}^t$. For machine i, we set $\ell_i^t = 1$ if the machine had a large load average (more than 0.5), and $\ell_i^t = 0$ if it had a small load average. The intent of this is to model the decision faced by a "user-friendly" background process that suspends its work if someone else is using the same machine.

We took the distance between the machines to be 0.1, indicating that 30 seconds of computation would be lost for

movement between machines. In research process migration systems, the time for a process to move is roughly proportional to its size. For a 100-KB process, the time is about a second [Esk90]. Our distance corresponds to large but reasonable memory usage.

Our simulations compared the performance of nine algorithms, including four simple control algorithms:

Uniform The algorithm picks a random machine and stays there.

Greedy After each trial the algorithm moves to the machine that incurred the least loss in that trial.

Least-Used After each trial the algorithm moves to the machine that has incurred the least total loss so far.

Recent The algorithm moves to the machine that has incurred the least loss over the last k trials.

We implemented Marking, Odd-Exponent (with $t = 3$), Thresh, and Share. We also tried Work-Function, an MTS algorithm analyzed by Borodin, Linial, and Saks [BLS92]. This is a deterministic algorithm that, whenever its state's reduced loss is pinned, moves to the state that is pinning it (for the uniform metric space this corresponds to the state with the least reduced loss). For each algorithm we found the best parameters for our data set; that is, we assumed that the programmer would optimize these parameters beforehand.

For each algorithm we determined the expected loss for the probability vectors they calculated. One valid criticism of using probabilistic algorithms for real problems is the variance between runs; so we also calculated the standard deviation over 200 trials of each algorithm. To get a feel of how each algorithm behaves, we included the expected number of moves.

This data is summarized in Table 1 where costs are given relative to the optimal off-line sequence, which suffered a loss of 15.9 and moved 39 times during the 1872 trials.

We also tried an inter-machine distance of 1.0. Table 2 summarizes these results. The optimal off-line sequence suffered a loss of 46.0 and moved 22 times during the 1872 trials.

Comparing these algorithms to the simpler control algorithms indicates that their added sophistication does indeed help. The numbers do not indicate whether MTS-based algorithms or weighted-expert algorithms are better for this problem. This specific experiments summarized here have the MTS algorithms performing mildly better, but other experiments do not show a general trend.

Most fascinating is how similarly these algorithms perform despite the differences in their techniques and backgrounds. The experiment supports the analysis of this paper, which illustrates how these separate techniques can work for both scenarios.

6 Summary

In this paper we consider the relationship between the Metrical Task System problem and the problem of tracking the best expert in the "decision-theoretic" experts setting. We give some general relationships, showing that any MTS algorithm also has a partitioning bound in the Experts-DTF setting based on its r-unfair competitive ratio in the uniform metric space. We also consider several specific existing algorithms, and introduce and analyze several new variations. Their bounds are summarized in Table 3.

Interestingly, algorithms of quite different styles — the "work-function-based" or "reduced-loss-based" algorithms designed for the MTS problem, and the multiplicative weight-updating schemes more common in the experts setting — yield similar bounds. In the MTS setting, the somewhat better theoretical bounds of Odd-Exponent may be offset by the relative simplicity, intuitiveness, and ease of implementation of Share.

There are many settings that combine the problem of picking the best decision/expert/state with a cost for frequently changing one's mind. The process migration problem is one for which we have performed some simple experiments. It would be interesting to see more generally what sorts of algorithms turn out to be best for other problems of this nature.

References

[BBBT97] Y. Bartal, A. Blum, C. Burch, and A. Tomkins. A polylog(n)-competitive algorithm for metrical task systems. In *Annual ACM Symposium on Theory of Computing*, 1997. To appear.

[BKRS92] A. Blum, H. Karloff, Y. Rabani, and M. Saks. A decomposition theorem and lower bounds for randomized server problems. In *Proc IEEE Symposium on Foundations of Computer Science*, pages 197–207, 1992.

[BLS92] A. Borodin, N. Linial, and M. Saks. An optimal online algorithm for metrical task systems. *J of the ACM*, 39(4):745–763, 1992.

[Chu94] T. Chung. Approximate methods for sequential decision making using expert advice. In *Proc Annu. ACM Workshop on Comput. Learning Theory*, pages 183–189. ACM Press, New York, NY, 1994.

[Esk90] M. Eskicioglu. Process migration in distributed systems: A comparitive survey. Technical Report TR 90-3, University of Alberta, January 1990.

[FKL+91] A. Fiat, R. Karp, M. Luby, L. McGeoch, D. Sleator, and N. Young. Competitive paging algorithms. *J of Algorithms*, 12:685–699, 1991.

[FS95] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, pages 23–37, 1995.

[HW95] M. Herbster and M. Warmuth. Tracking the best expert. In *Proc International Conference on Machine Learning*, pages 286–294. Morgan Kaufmann, 1995.

[IS95] S. Irani and S. Seiden. Randomized algorithms for metrical task systems. In *Intl. Workshop on Algorithms and Data Structures*, pages 159–170, 1995.

[LW94] N. Littlestone and M. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

[Sei96] S. Seiden. Unfair problems and randomized algorithms for metrical task systems. Manuscript, April 1996.

algorithm	parameter setting	cost ratio	std dev	expected moves
Uniform		96.523	12.78	0.00
Greedy		29.925	0.00	659.97
Least-Used		67.076	0.00	6.97
Recent	$k = 11$	6.724	0.00	70.97
Work-Function	$r = 4.0$	4.881	0.00	57.97
Marking	$r = 0.5$	5.308	0.27	75.50
Odd-Exponent	$t = 3, r = 1.0$	5.017	8.90	59.40
Thresh	$\beta = 6.5 \times 10^{-12}, \alpha = 1 \times 10^{-10}$	5.826	0.25	45.72
Share	$\beta = 1.5 \times 10^{-8}, \alpha = 1 \times 10^{-10}$	5.769	0.23	50.44

Table 1: Performance relative to optimal off-line sequence ($d = 0.1$) on process migration data

algorithm	parameter setting	cost ratio	std dev	expected moves
Uniform		33.363	4.60	0.00
Greedy		23.256	0.00	659.97
Least-Used		23.321	0.00	6.97
Recent	$k = 29$	3.408	0.00	41.97
Work-Function	$r = 0.617$	2.691	0.00	39.97
Marking	$r = 0.071$	3.248	0.16	74.03
Odd-Exponent	$t = 3, r = 1.028$	2.795	2.37	51.42
Thresh	$\beta = 6.5 \times 10^{-10}, \alpha = 1 \times 10^{-8}$	2.908	0.14	45.72
Share	$\beta = 2.6 \times 10^{-6}, \alpha = 1 \times 10^{-10}$	2.935	0.15	44.21

Table 2: Performance relative to optimal off-line sequence ($d = 1.0$) on process migration data

Algorithm	competitive ratio		partitioning bound	
Linear ($n = 2$)	$1 + 1/r$	[BKRS92]	$(1 + 1/2r)L + (r + 1/2)k$	(Th 3)
Marking	$H_n(1 + 1/r)$	[BLS92]	$H_n(1 + 1/r)L + H_n(r + 1)k$	(Cor 4)
Odd-Exponent	$1 + 2e \ln(n)/r$	[BBBT97]	$(1 + 1/c)L + (2e(c + 1) \ln n)k$	(Cor 6)
Thresh	unbounded		$\left(\frac{\ln(1/\beta)}{(1-\beta)(1-\alpha)}\right) L + \left(\frac{\ln(n/\beta\alpha)}{(1-\beta)(1-\alpha)}\right) k$	(Th 7)
Share	$1 + 8 \ln((n + 2)(2r + 1))/r$	(Th 9)	$\left(\frac{\ln(1/\beta)}{(1-\beta)(1-\alpha)}\right) L + \left(\frac{\ln((n+2)/\alpha)}{(1-\beta)(1-\alpha)}\right) k$	(Th 8)

Table 3: Summary of our theoretical results