Mixing predictions for online metric algorithms

Antonios Antoniadis [*1] **Christian Coester** [*2] **Marek Eliáš** [*3] **Adam Polak** [*4] **Bertrand Simon** [*5]

Abstract

A major technique in learning-augmented online algorithms is combining multiple algorithms or predictors. Since the performance of each predictor may vary over time, it is desirable to use not the single best predictor as a benchmark, but rather a dynamic combination which follows different predictors at different times. We design algorithms that combine predictions and are competitive against such dynamic combinations for a wide class of online problems, namely, metrical task systems. Against the best (in hindsight) unconstrained combination of ℓ predictors, we obtain a competitive ratio of $O(\ell^2)$, and show that this is best possible. However, for a benchmark with slightly constrained number of switches between different predictors, we can get a $(1+\epsilon)$-competitive algorithm. Moreover, our algorithms can be adapted to access predictors in a bandit-like fashion, querying only one predictor at a time.

1. Introduction

Motivated by the power of machine-learned predictions, the field of learning-augmented algorithms has been growing rapidly in recent years. In the classical field of online algorithms, an input sequence is revealed to an algorithm over time and it is assumed that at all times, no information about the future part of the input is available. In contrast, a learning-augmented algorithm additionally has access to *predictions* (e.g., machine-learned) related to the future input. These predictions may be inaccurate, so a challenge is to simultaneously utilize high-quality predictions to their best advantage while at the same time avoiding to be misled by erroneous predictions.

An important technique in the field of learning-augmented algorithms is the method of combining multiple algorithms into a single hybrid algorithm that leverages the advantages of all individual algorithms. The basic idea goes back to several decades before the area of learning-augmented algorithms was born and also has applications, for example, in pure online algorithms: Fiat et al. (1990) defined a MIN operator on algorithms for the k-server problem that combines several algorithms into one whose cost matches the best of them up to a constant factor, and they used this technique to obtain the first competitive algorithm for the k-server problem.

In learning-augmented algorithms, similar combination techniques are employed for several purposes. Firstly, they are frequently used to make algorithms robust against prediction errors by combining an algorithm that mostly follows predictions with a classical online algorithm that ignores predictions (see, e.g., (Lykouris & Vassilvitskii, 2021; Purohit et al., 2018; Rohatgi, 2020; Antoniadis et al., 2020; Wei, 2020; Bamas et al., 2020; Bansal et al., 2022)). In fact, one might argue that almost all algorithms that utilize predictions while being robust to their error are at least implicitly a kind of combination of two algorithms. A second purpose, as employed in (Antoniadis et al., 2021), is to combine several differently parameterized versions of the same algorithm in order to perform nearly as well as the version with the best parameter choice. These aforementioned works, like the majority of research in learning-augmented algorithms, focus on settings where a single predictor provides suggestions to the algorithm.

However, a third and perhaps the most relevant application of combining several algorithms in the learning-augmented realm is to be able to deal with multiple predictors. In practice it is often the case that several predictors are available, but they produce potentially conflicting advice; for example, there may be different ML models based on different methods or tailored to specific scenarios, or several human experts with contrary opinions. Since it is not clear a priori which of the predictors will be most reliable for the instance at hand, this creates the complication of deciding how to choose between the predictors. Research on learning-augmented algorithms with multiple predictions

[*]Equal contribution [1]University of Twente, The Netherlands [2]University of Oxford, Oxford, United Kingdom [3]Bocconi University, Milan, Italy [4]Max Planck Institute for Informatics, Saarbrucken, Germany [5]CNRS / IN2P3 Computing Center, Villeurbanne, France. Correspondence to: Antonios Antoniadis <a.antoniadis@utwente.nl>, Christian Coester <christian.coester@cs.ox.ac.uk>, Marek Eliáš <marek.elias@unibocconi.it>, Adam Polak <apolak@mpi-inf.mpg.de>, Bertrand Simon <bertrand.simon@cnrs.fr>.

was initiated by Gollapudi & Panigrahi (2019) for the ski rental problem, and subsequently also considered for additional problems such as multi-shop ski rental (Wang et al., 2020), facility location (Almanza et al., 2021), matching, load balancing and non-clairvoyant scheduling (Dinitz et al., 2022). For the objective of regret minimization, the case of multiple predictions was studied for online linear optimization (Bhaskara et al., 2020) as well as caching (Emek et al., 2021). Recently, Anand et al. (2022) designed a generic framework for online covering problems with multiple predictions, which they successfully applied to the problems of set cover, weighted caching and facility location.

A particularly interesting aspect of the work of Anand et al. (2022) is that the performance of their algorithm is comparable not only to the best individual predictor, but even to the best dynamic combination of predictors. This property is especially valuable on instances which are composed of several parts with different properties, as some predictor may be of high quality for certain sections of the input, but inferior to other predictors otherwise.

This raises the question of whether similar guarantees are also achievable for other problems. Anand et al. (2022) mention the k-server problem as a specific problem for which this would be interesting.

Here, our goal is to obtain generic methods for combining multiple predictors/algorithms applicable to a wide range of problems. To this end, we consider the class of *metrical task systems (MTS)*. MTS was introduced by (Borodin et al., 1992) as a wide class of online problems, containing as special cases many other fundamental online problems such as k-server, caching, convex body/function chasing, layered graph traversal, dynamic power management etc. Thus, our results obtained for MTS directly translate to all these other problems as well.

We study this class of problems in a variety of settings: Against the best (in hindsight) dynamic combination of ℓ predictors, we obtain a competitive ratio of $O(\ell^2)$ and show that this is the best possible. This follows, essentially, from a reduction to the layered graph traversal problem. The aforementioned result allows the benchmark offline combination to switch between the ℓ predictors arbitrarily often. However, for more structured instances (for example, imagine an instance composed of blocks with different patterns, and different predictors specialized on these patterns), it is reasonable to assume that an optimal combination of predictors would not switch between them too often. We therefore consider the question whether a better performance is achievable under such an assumption. Indeed, against a dynamic combination benchmark that switches between the different predictors a moderately limited number of times, we achieve a $(1+\epsilon)$-competitive algorithm.

Since querying predictors may be costly (Im et al., 2022; Emek et al., 2021), we also consider a setting where the learning-augmented algorithm can consult only one predictor per time step (similar to multi-armed bandits). We show that very similar guarantees can be achieved also for this setting.

1.1. Preliminaries

Metrical Task Systems. In Metrical Task Systems (MTS), we are given a metric space (M, d), whose points are called *states*. An algorithm starts in some initial state $s_0 \in M$. At each time $t = 1, 2, \dots, T$, a task appears, specified by some cost function $c_t \colon M \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ that assigns to each state the cost of serving the task in that state. In response, the algorithm chooses a state $s_t \in M$, paying *movement cost* $d(s_{t-1}, s_t)$ and *service cost* $c_t(s_t)$. We emphasize that s_t can be chosen *after* c_t is known, but before c_{t+1} is revealed.

***k*-server.** In the k-server problem, we are given a metric space (M, d), and k servers are located at points of M. At each time $t = 1, 2, \dots, T$, a point $r_t \in M$ is requested, and an algorithm must choose one of the servers to move to r_t. The cost is the total distance travelled by servers. Note that k-server is an MTS in the metric space of server configurations.[1]

Competitive ratio. An algorithm A for an online minimization problem is called ρ-competitive if

$$\mathrm{cost}(A) \leq \rho \cdot \mathrm{OPT} + c \qquad (1)$$

for every instance of the problem, where $\mathrm{cost}(A)$ is the cost of A on the instance (or the expected cost, if A is randomized), OPT is the optimal (offline) cost, and c is a constant independent of the input sequence. If we replace OPT by some other benchmark B, we also say that A is *ρ-competitive against B*. The minimal ρ satisfying (1) is also called the *competitive ratio*.

Prediction setup. We consider the setting where there are ℓ predictors $P_1, \dots, P_\ell$. At each time t, predictor P_i produces a suggestion of a state $\varphi_{it} \in M$ where the algorithm should go. Note that we may think of each P_i itself as an algorithm to serve the request sequence. The case $\ell = 1$ of a single predictor was studied in (Antoniadis et al., 2020).

To evaluate the performance of our algorithms, we consider as benchmark algorithms the best dynamic combination of the predictors. We write DYN for the cost of the best (offline) algorithm that is in one of the predicted states at each time step:

[1] I.e., k-server in M can be cast as an MTS by taking the set of k-server configurations (i.e., size-k-subsets of M) as the metric space for MTS. Then c_t assigns cost 0 to configurations containing r_t and cost ∞ to other configurations.

$$\mathrm{DYN} := \min_{\substack{s_1,\dots,s_T:\\ s_t\in\{\varphi_{1t},\dots,\varphi_{\ell t}\}}} \sum_{t=1}^{T} d(s_{t-1}, s_t) + c_t(s_t)$$

If $s_t = \varphi_{it}$, we say that the algorithm *follows* P_i at time t. We define $\mathrm{DYN}^{\leq m}$ similarly to DYN, but for an offline algorithm that switches the predictor that it is following at most m times.

For the k-server problem, note that φ_{it} is a configuration (i.e., a set) of k points. Here, it is natural to consider predictors that are *lazy*, i.e., they move a server only to serve a request; formally, $\varphi_{it} \subseteq \varphi_{i,t-1} \cup \{r_t\}$. In this case, the sequence of predictions produced by P_i can also be encoded by specifying for each time t only the name of the server that should serve the current request. This suggests an alternative definition of a dynamic combination for the k-server problem: We write $\widetilde{\mathrm{DYN}}$ for the cost of the best offline algorithm that serves each request r_t using a server named by *any* of the predictors at time t.

Full access and bandit access. We define two types of learning-augmented algorithms for MTS, depending on the type of access they have to the predictors. Note that in both cases, they see the input cost function c_t. In the *full access* model, the algorithm receives at each time t as additional input the ordered tuple $(\varphi_{1t}, \varphi_{2t}, \dots, \varphi_{\ell t})$. In the *Bandit access* model, the algorithm chooses some $i_t \in \{1, \dots, \ell\}$ at time t and only observes the state $\varphi_{i_t t}$ and the (movement + service) cost paid by P_{i_t} at time step t.[2] For our algorithms, it does not matter whether i_t is chosen before or after the cost function c_t is observed. In all cases, the learning-augmented algorithm has to choose its own state s_t only *after* observing the full cost function c_t as well as the predicted state(s) for time t.

1.2. Our results

We begin by stating a negative result concerning the benchmark $\widetilde{\mathrm{DYN}}$ for the k-server problem, suggesting that this benchmark is too strong, even if there are only two predictors.

Theorem 1.1. *For the k-server problem on the line metric with full access to two predictors, every deterministic (resp. randomized) learning-augmented algorithm has competitive ratio at least k (resp. $\Omega(\log k)$) against $\widetilde{\mathrm{DYN}}$.*

Since k is the exact deterministic competitive ratio and $\Omega(\log k)$ is the best known lower bound on the randomized competitive ratio of k-server on the line metric *without predictions* (Manasse et al., 1990; Chrobak et al., 1991; Bubeck et al., 2022a), predictions do not seem useful against this benchmark. We therefore dismiss this benchmark for the remainder.

For the benchmark DYN, we obtain the following result for any MTS (and therefore also the k-server problem) by a reduction to the layered graph traversal problem:

Theorem 1.2. *For any MTS problem with full access to ℓ predictors, there is an $O(\ell^2)$-competitive randomized algorithm against* DYN.

A similar connection to layered graph traversal (or the equivalent metrical service systems problem) yields the following matching lower bound:

Theorem 1.3. *There exist instances of MTS and k-server where no learning-augmented algorithm with full access to ℓ predictors can achieve a competitive ratio better than $\Omega(\ell^2)$ against* DYN.

We remark that the algorithm from Theorem 1.2 can be made robust against prediction errors, so that it achieves a cost of at most $O\left(\min\{\rho \cdot \mathrm{OPT}, \ell^2 \cdot \mathrm{DYN}\}\right)$, where ρ is the best competitive ratio of the given MTS problem in the setting without predictions. An analogous robustification is possible for all of our algorithms, and we will not mention it explicitly. For details, we refer to (Antoniadis et al., 2020).

For the bandit-access model, we note that previous results by Emek et al. (2009) allow to transfer the guarantees of Theorem 1.2, losing only a factor ℓ:

Theorem 1.4. *For any MTS problem with bandit access to ℓ predictors, there is an $O(\ell^3)$-competitive randomized algorithm against* DYN.

As argued above, a more realistic benchmark might be a dynamic combination whose number of switches between predictors is somewhat bounded. In analogy to results about tracking best experts in online learning (Cesa-Bianchi & Lugosi, 2006), one might expect a competitive ratio of $1+\epsilon$ against a dynamic combination that switches at most $f(\epsilon)T$ times. However, T is not an adequate quantity to express the length of an MTS instance, as dummy tasks of cost 0 can artificially inflate the length. Instead, we use the value of DYN to scale our results based on the meaningful length of the instance.

Theorem 1.5. *For any MTS with full access to ℓ predictors and any $\epsilon > 0$, there is a $(1+\epsilon)^2$-competitive randomized algorithm against $\mathrm{DYN}^{\leq m}$, where $m = \Omega\left(\frac{\epsilon^2}{\log \ell} \cdot \frac{\mathrm{DYN}}{D}\right)$ and D is the diameter of the underlying metric space.*

Theorem 1.5 generalizes a result of Blum & Burch (2000), who showed that a competitive rato of $1+\epsilon$ is achievable against $\mathrm{DYN}^{\leq 0}$ (i.e., a static benchmark that does not switch at all).

[2] In the full access model, the learning-augmented algorithm also knows the cost of each algorithm P_i for all time steps, as it can be deduced from its state at the current and previous time step.

We remark that it is not necessary for our algorithm to know the value of DYN. Note that the result also holds for large ϵ (in which case ϵ would be the dominant term in $(1+\epsilon)^2$). The following result shows that the bound obtained is asymptotically optimal for large ϵ, and the dependency on DYN, D and ℓ cannot be improved.

Theorem 1.6. *For any $\epsilon > 0$, there exists an MTS with full access to ℓ predictors on which no randomized algorithm can be $(1+\epsilon)^2$-competitive against $\mathrm{DYN}^{\leq m}$ if $m \geq \frac{6(1+\epsilon)^2}{\log \ell} \frac{\mathrm{DYN}}{D}$.*

For the Bandit access model with a limited number of switches, we obtain the following result.

Theorem 1.7. *For any MTS with Bandit access to ℓ predictors and any $\epsilon > 0$, there is a $(1+\epsilon)^3$-competitive randomized algorithm against $\mathrm{DYN}^{\leq m}$, where*

$$m = \Omega\left(\frac{\epsilon^3/\log(2+\epsilon^{-1})}{\ell \log \ell} \cdot \frac{\mathrm{DYN}}{D}\right)$$

and D is the diameter of the underlying metric space.

1.3. Organization

The remainder of the paper is organized as follows. In Section 2, we study upper and lower bounds on the achievable competitive ratio against DYN and $\widetilde{\mathrm{DYN}}$ with full access to predictors. In Sections 3 and 4, we respectively show positive and negative results when using $\mathrm{DYN}^{\leq m}$ as a benchmark. Finally, we focus on the bandit access setting in Section 5.

2. Unbounded number of switches

The goal of this section is to show tight bounds against DYN as well as a lower bound for k-server against $\widetilde{\mathrm{DYN}}$ (deferred to Appendix C in the full access model.

We start by showing that in the full access model, the competitive ratio against DYN (with unlimited number of switches) is $O(\ell^2)$ and $\Omega(\ell^2)$, proving Theorems 1.2 and 1.3.

The problem of combining predictors $P_1, \ldots, P_\ell$ on an MTS instance (ℓ-MTS for short) can be formulated as a classical MTS on the same underlying metric space: it is enough to modify the losses: $\ell'_t(s) := \ell_t(s)$ if s is a state of some predictor and $\ell'_t(s) := +\infty$ otherwise. This formulation does not yet make the problem easier: the underlying metric space remains the same with the same number of points n, seemingly keeping the complexity of the problem the same as solving the input instance directly. However, having finite loss only on at most ℓ states at a time allow us to reduce this problem to a more structured variant of MTS called ℓ-width *layered graph traversal (LGT)*. We will show this in a similar way to a reduction from metrical service systems to LGT from (Fiat et al., 1998).

In layered graph traversal (LGT) we are given a graph with non-negative edge weights, and a searcher that starts at a designated vertex s. The graph has the property that its vertices can be partitioned into layers $L_0 := \{s\}, L_1, L_2 \ldots$ such that any edge connects vertices of two consecutive layers. In ℓ-width LGT, one has $|L_t| \leq \ell$ for all t. The problem is online, meaning that the searcher is only aware of the edges (and corresponding weights) adjacent to the layers visited so far. Each traversal of an edge by the searcher incurs a cost equal to the weight of that edge. The goal is to move the searcher along the edges to a target vertex in the last layer. The cost is the distance travelled by the searcher.

It was shown recently that ℓ-width LGT admits an $O(\ell^2)$-competitive randomized algorithm (Bubeck et al., 2022b).

Proof of Theorem 1.2. Consider an instance I of ℓ-MTS. We can construct a corresponding instance I' of ℓ-width layered graph traversal as follows. Every layer L_t, $t \geq 1$ in I' consists of exactly ℓ vertices $v_{1t}, v_{2t}, \ldots v_{\ell t}$ where intuitively vertex v_{it} corresponds to the state φ_{it} of predictor P_i at time t. The edges between any two consecutive layers form a complete bipartite graph, where the weight of edge $(v_{i,t-1}, v_{jt})$ is set to $d(\varphi_{i,t-1}, \varphi_{jt}) + c_t(\varphi_{jt})$. Finally, all vertices of the last layer L_T constructed in this way are connected to single target vertex in layer L_{T+1} with edges of weight 0.

It can be easily verified that I' is a feasible ℓ-width layered graph traversal instance. Furthermore any solution to I' can be naturally (and in an online-fashion) transformed into a corresponding solution for I: If the searcher in I' moves to vertex v_{it} when layer L_t is revealed, then the corresponding t'th request in I is served in state φ_{it}. Note that by construction the costs of the two solutions are exactly the same, as going back to a previous layer is never beneficial because d is a metric. Similarly one can apply the opposite transformation to the offline solution achieving cost DYN for instance I to obtain an offline solution of the same cost for instance I' for ℓ-width LGT.

The result follows, by the $O(\ell^2)$-competitive algorithm for ℓ-width LGT by Bubeck et al. (2022b). □

On the other hand, our lower bound in Theorem 1.3 can be derived via the *Metrical Server Systems (MSS)* problem (Chrobak & Larmore, 1991), which is in fact equivalent to LGT. In Metrical Server Systems (MSS) (Chrobak & Larmore, 1991), a server can move between the points of a metric space. In each round it is presented with a request, which consists of w points of the metric. In response, the server has to move to one of these w points. The goal is to minimize the total distance traversed by the server.

The proof of Theorem 1.3 follows by observing the relationship between MSS and respectively ℓ-MTS and k-server, see Appendix B.

3. Limited number of switches

Consider predictors $P_1, \dots, P_\ell$ for some MTS instance I with diameter D. In order to construct an algorithm for combining these predictors, we create a new MTS instance U on a uniform metric space with ℓ points, each corresponding to one of the predictors. At each time step, we calculate, for each $i = 1, \dots, \ell$, the cost $f_t(P_i)$ incurred by predictor P_i on instance I at time t, which includes both the movement and service cost of P_i, and issue the cost function c_t^U such that $c_t^U(i) = \frac{1}{D} f_t(P_i)$.

A solution to the instance U produced by some algorithm $\bar{A}$ can be translated to a combination of the predictors: whenever $\bar{A}$ resides at state i, we move to the current state of the predictor P_i. Service costs in U correspond to the scaled costs of the individual predictors. Therefore, if $\bar{A}$ always resides in state i, its total cost will be $\frac{1}{D}$ times the total cost of P_i serving the instance I. However, this translation does not preserve switching costs: neither for our algorithm nor for the optimal combination. While moving from i to j in instance U always costs 1, switching from P_i to P_j may cost anything between 0 and D. Algorithm 1 summarizes this translation.

Algorithm 1: $\textsc{Combine}_{\bar{A}}(P_1, \dots, P_\ell)$

$D :=$ diameter of the space;
foreach $t = 1, \dots, T$ **do**
  $c_t^U(i) = D^{-1} f_t(P_i)$ for each $i = 1, \dots, \ell$;
  $i_t :=$ state of $\bar{A}$ after seeing $c_1^U, \dots, c_t^U$;
  Move to the state of P_{i_t};

We choose algorithm $\bar{A}$ based on the following performance metric.

Definition 3.1 (Unfair competitive ratio)**.** Let $r > 0$. We say that an MTS algorithm A is r-unfair competitive if there is a constant $\alpha \geq 0$ such that for any instance the cost incurred by A is

$$\begin{aligned}\operatorname{cost}(A) &= \sum_{t=1}^{T} (c_t(x_t) + d(x_{t-1}, x_t)) \\ &\leq R \cdot \min_{y\colon y_0 = x_0} \{\sum_{t=1}^{T} (c_t(y_t) + r d(y_{t-1}, y_t))\} + \alpha,\end{aligned}$$

where x is the solution produced by the algorithm and the minimum in the right-hand side is the cost of the optimal solution whose movement costs are scaled by factor r. We call R the r-unfair competitive ratio of A.

Unfair ratios are usually considered with $r \leq 1$, i.e., the reference optimal solution pays cheaper costs for its movement than the algorithm, since this setting is important in the design of general algorithms for MTS (Fiat & Mendel, 2003; Bubeck et al., 2021). In our case, we are trying to prevent the optimum solution from moving too much, that's the intuition due to which we are interested in $r > 1$. The algorithm $\textsc{OddExponent}$ by Bartal et al. (1997) achieves the following bound also with $r \geq 1$. We denote by $r(\epsilon)$ the minimal r such that there is an algorithm with r-unfair competitive ratio $1 + \epsilon$ for the ℓ-point uniform metric. A description of $\textsc{OddExponent}$ can be found in Appendix A.1.

Proposition 3.2 (Bartal et al. (1997))**.** *Given r, there is an algorithm for the ℓ-point uniform metric space with r-unfair competitive ratio $1 + \frac{1}{r} 2e \ln \ell$. This gives $r(\epsilon) = O(\epsilon^{-1} \ln \ell)$.*

The following lemma relates the cost of $\textsc{Combine}_{\bar{A}}(P_1, \dots, P_\ell)$ to the cost of an optimal combination which has to pay a fixed large cost for every switch between two predictors.

Lemma 3.3. *Let $\bar{A}$ be an algorithm for uniform MTS and $r_{\bar{A}}(\epsilon)$ be such that the $r_{\bar{A}}(\epsilon)$-unfair competitive ratio of $\bar{A}$ is $(1 + \epsilon)$, for some $\epsilon > 0$. Let DYN_ρ denote the optimal cost of a combination of predictors $P_1, \dots, P_\ell$ which pays $\rho = 2D r_{\bar{A}}(\epsilon)$ for each switch between two predictors. Then $\textsc{Combine}_{\bar{A}}(P_1, \dots, P_\ell)$ is $(1+\epsilon)$-competitive with respect to DYN_ρ.*

Proof. Denote $\mathrm{OPT}^U_{r_{\bar{A}}(\epsilon)}$ the cost of the optimum solution for U which pays $r_{\bar{A}}(\epsilon)$ instead of 1 for each movement. We know that the cost of $\bar{A}$ is at most $(1+\epsilon)\,\mathrm{OPT}^U_{r_{\bar{A}}(\epsilon)} + \alpha$ for some constant α.

Now, $\textsc{Combine}_{\bar{A}}$ pays $f_t(P_i)$ when following P_i or, if there was a switch, at most $D + f_t(P_i)$. In the same situation, $\bar{A}$ pays $D^{-1} f_t(P_i)$ and $1 + D^{-1} f_t(P_i)$ respectively. Therefore, the total cost of $\textsc{Combine}_{\bar{A}}$ is at most

$$\sum_{t=1}^{T} D \operatorname{cost}_t(\bar{A}) \leq D \cdot \big((1+\epsilon)\,\mathrm{OPT}^U_{r_{\bar{A}}(\epsilon)} + \alpha\big).$$

To show that $\mathrm{OPT}^U_{r_{\bar{A}}(\epsilon)} \leq \frac{1}{D} \mathrm{DYN}_\rho$, we translate DYN_ρ into a (possibly suboptimal) solution on instance U as follows: If DYN_ρ follows P_i at time t and pays cost $f_t(P_i)$, we stay at state i in U and pay cost $\frac{1}{D} f_t(P_i)$. Otherwise, DYN_ρ switches from P_j to P_i at time t and pays cost $2D r_{\bar{A}}(\epsilon) + srv(P_i)$, where $srv(P_i)$ denotes the service cost paid by P_i. We move from state j to i in U and pay $r(\epsilon) + \frac{1}{D} f_t(P_i) \leq r(\epsilon) + 1 + \frac{1}{D} srv(P_i)$, because the moving cost of P_i is at most D. In both cases, the cost incurred by DYN_ρ was D times larger than the constructed solution on U, implying $\mathrm{OPT}^U_{r_{\bar{A}}(\epsilon)} \leq \frac{1}{D} \mathrm{DYN}_\rho$. $\square$

Theorem 1.5 follows from the following lemma translating the competitive ratio with respect to DYN_ρ to a competitive ratio with respect to $\mathrm{DYN}^{\le m}$.

Lemma 3.4. *Let $\epsilon > 0$ and $\rho > 0$. If an algorithm A is $(1+\epsilon)$-competitive against DYN_ρ, then it is $(1+\epsilon)^2$-competitive against $\mathrm{DYN}^{\le m}$ for any $m \le \epsilon \,\mathrm{DYN}/\rho$.*

Proof. Let us denote α such that $\mathrm{cost}(A) \le (1+\epsilon)\,\mathrm{DYN}_\rho + \alpha$. Relating its cost to $\mathrm{DYN}^{\le m}$ for any $m \le \epsilon\,\mathrm{DYN}/\rho$, we have

$$\begin{aligned}\mathrm{cost}(A) &\le \big((1+\epsilon)\,\mathrm{DYN}_\rho + \alpha\big)\\ &\le (1+\epsilon)(\mathrm{DYN}^{\le m} + \epsilon\,\mathrm{DYN}) + \alpha\\ &\le (1+\epsilon)^2\,\mathrm{DYN}^{\le m} + \alpha,\end{aligned}$$

because $\mathrm{DYN}_\rho \le \mathrm{DYN}^{\le m} + m\rho$ and $\mathrm{DYN} \le \mathrm{DYN}^{\le m}$. □

Using Lemma 3.3 and choosing $\bar{A}$ from Proposition 3.2, we get that $\textsc{Combine}_{\bar{A}}$ is $(1+\epsilon)^2$-competitive with respect to $\mathrm{DYN}^{\le m}$ whenever $m \le \frac{\epsilon^2}{4De\ln\ell}\,\mathrm{DYN}$, as claimed by Theorem 1.5.

4. Hardness for limited number of switches

In this section we show Theorem 1.6, stating that the bound on the maximum number of allowed switches m given by Theorem 1.5 is tight up to a constant factor, for fixed ϵ. In particular, the asymptotic dependence on ℓ, D, and DYN is optimal.

The randomized construction we use in this section is inspired by the classical coupon collector lower bound for MTS (Borodin et al., 1992). We consider a uniform[3] metric space with ℓ points. There are also ℓ predictors, the i-th of them predicting to always stay at point i. Let $\sigma_1, \dots, \sigma_T$ be T independent random variables, each drawn uniformly from the metric space. Let $\alpha \in (0,1]$ be a parameter. At time step t, the cost function is

$$c_t(x) = \begin{cases} 1 & \text{if } x = \sigma_t, \\ \alpha/\ell & \text{if } x \ne \sigma_t. \end{cases}$$

In each step, any online algorithm (even given access to the above predictors, whose predictions are independent from the random instance) has expected cost of at least $1/\ell$, since with probability $1/\ell$ the random point σ_t falls on the old state of the algorithm, and the algorithm either moves and incurs moving cost 1 or stays and incurs service cost 1. After T steps, the expected total cost of an algorithm A is at least $\mathbb{E}[\mathrm{cost}(A)] \ge T/\ell$.

[3] I.e., the distance between any two different points is 1.

Clearly, $\mathrm{DYN} \ge T\alpha/\ell$. Let $m = \frac{2\,\mathrm{DYN}}{\alpha\ln\ell} \ge \frac{4T}{\ell\ln\ell}$. We will upper bound the expected value of $\mathrm{DYN}^{\le m}$ by considering the following offline strategy. Whenever σ_t hits the currently followed predictor, switch to the predictor that will be hit furthest in the future (i.e., akin to Belady's rule for the caching problem), unless the switching budget m has already run out.

Let X be the random variable denoting the number of steps from a given switch until the next switch. By a coupon-collector analysis, $\mathbb{E}[X] = \sum_{i=1}^{\ell-1}\mathbb{E}[\mathrm{Geo}(i/\ell)] = \sum_{i=1}^{\ell-1}\ell/i > \ell\ln\ell$, where $\mathrm{Geo}(p)$ denotes a geometrically distributed random variable with success probability p. Moreover,

$$\mathrm{Var}(X) = \sum_{i=1}^{\ell-1}\mathrm{Var}(\mathrm{Geo}(i/\ell)) = \sum_{i=1}^{\ell-1}\frac{1-\frac{i}{\ell}}{\left(\frac{i}{\ell}\right)^2} \le \frac{\pi^2}{6}\cdot\ell^2.$$

Let Y be the random variable denoting the expected number of switches until time T when ignoring the upper bound m. The central limit theorem for renewal processes shows that

$$\lim_{T\to\infty}\frac{\mathbb{E}[Y]}{T} = \frac{1}{\mathbb{E}[X]} < 1/(\ell\ln\ell) \quad \text{and}$$
$$\lim_{T\to\infty}\frac{\mathrm{Var}[Y]}{T} = \frac{\mathrm{Var}[X]}{\mathbb{E}[X]^3} < 1.$$

Therefore, for large enough T,

$$\mathbb{E}[Y] \le \frac{T}{\ell\ln\ell} \quad \text{and} \quad \mathrm{Var}[Y] \le T.$$

For large enough T, the switching budget $m \ge \frac{2T}{\ell\ln\ell}$ is at least $\mathbb{E}[Y] + \sqrt{T}/\ell\ln\ell\cdot\sqrt{\mathrm{Var}(Y)}$. Hence, by Chebyshev's inequality, the probability of running out of the switching budget can be upper bounded by $P(Y > m) \le \ell^2\ln^2\ell/T$, and in that case the expected total cost of following a fixed predictor can be upper bounded by $T(1+\alpha)/\ell$. In the event the strategy does not run out of the switching budget, the total service cost is $T\alpha/\ell$ and the expected movement cost is at most $T/(\ell\ln\ell)$. Summing up,

$$\mathbb{E}[\mathrm{DYN}^{\le m}] \le T\alpha/\ell + T/(\ell\ln\ell) + P(Y > m)T(1+\alpha)/\ell.$$

For ℓ and T large enough, we get $\mathbb{E}[\mathrm{DYN}^{\le m}] < 3\alpha T/\ell$.

Since for any online algorithm A (with predictions) we have $\mathbb{E}[\mathrm{cost}(A)] \ge T/\ell$, we conclude that for any constant c there exists T large enough such that $\mathbb{E}[\mathrm{cost}(A)] \ge \mathbb{E}[DYN^{\le m}]/(3\alpha) + c$ for the random request sequence (and hence there also exists a deterministic sequence for which the inequality holds). We conclude that no algorithm can be better than $(1/3\alpha)$-competitive against a combination of predictors that allows $\frac{2}{\alpha\ln\ell}$ DYN switches, on a metric space with diameter $D = 1$. The generalization to arbitrary values of D can be made by scaling distance and service costs by a factor D and replacing DYN by $^{\text{DYN}}/_{D}$ in the definition on m. Setting $\alpha = 1/(3\cdot(1+\epsilon)^2)$ yields $m = \frac{6(1+\epsilon)^2}{\ln \ell}\frac{\text{DYN}}{D}$, proving Theorem 1.6.

5. Bandit access to predictors

In this section, we focus on a more restrictive setting inspired by the multi-armed bandit model, and motivated by the fact that querying many predictors may be expensive: at each time t, the algorithm still has access to the full cost function c_t of the original MTS instance, but it is able to query the state of only one predictor. Only after selecting which predictor j to query at time t, the algorithm is aware of its state φ_{jt} and of its (movement + service) cost $f_t(j)$ incurred at this time step. Then, the algorithm chooses its own state, which does not necessarily have to be φ_{jt}.

5.1. Unbounded number of switches

We propose an algorithm which queries the predictors round-robin, i.e., for each i, predictor P_i is queried at time steps $i, \ell+i, 2\ell+i, 3\ell+i, \dots$. It is known how to convert these queries of P_i into an explicit algorithm P'_i whose performance is at most $O(\ell)$-times worse than P_i.

Proposition 5.1 (Emek et al. (2009)). *There is an algorithm which, receiving knowledge of the state of an MTS algorithm P every ℓ time steps, is $O(\ell)$-competitive against P.*

So, we apply Theorem 1.2 to $P'_1, \dots, P'_\ell$, getting a combination which is $O(\ell^2)$ competitive with respect to the best combination of $P'_1, \dots, P'_\ell$. By the above proposition, the best combination of $P'_1, \dots, P'_\ell$ is at most $O(\ell)$-times worse than the best combination of $P_1, \dots, P_\ell$. Our competitive ratio is then $O(\ell^3)$, proving Theorem 1.4.

5.2. Limited number of switches

We consider an MTS instance of finite diameter D. We assume that, at each time step t, there is a state x such that $c_t(x) = 0$. This is without loss of generality: we can modify the cost function by subtracting $\min_x\{c_t(x)\}$ from the cost of each state at time t. Since this discounts the cost of all algorithms (including the benchmark) by the same additive quantity, the competitive ratio on the original instance is no larger than on the modified instance. We can further assume that $f_t(i) \leq 2D$ for each i and t: if this is not the case, we move to the state with cost 0 (guaranteed by the assumption above), serve the task there and move back to φ_{it}, paying at most $2D$ in total.

Let $\bar{A}$ be an algorithm for unfair MTS on uniform metric spaces. Algorithm 2 for the bandit access model creates a suitable MTS instance on the uniform metric space of size ℓ and uses $\bar{A}$ to choose which predictor a_t to follow, moving to state $b_t = \varphi_{a_t t}$. However, with a small probability γ, it does not query the state of a_t, querying a random predictor instead – we call this an exploration step. This is a common technique in multi-armed bandits, see (Slivkins, 2019) for instance. During an exploration step at time t, it makes greedy steps from b_{t-1} to a state g_t. Once serving the cost function at g_t, it returns back to $b_t = b_{t-1}$. The algorithm is described in Algorithm 2. It requires a parameter $0 < \gamma < 1/4$ which denotes the exploration rate.

Algorithm 2: BANDITCOMBINE $(P_1, \dots, P_\ell)$

Select $X \subseteq [T]$ by choosing each $t \in [T]$ independently with probability γ;
For each $t \in X$: choose $i_t \in \{1, \dots, \ell\}$ uniformly at random;
for $t = 1, \dots, T$ **do**
  if $t \in X$ **then** `/* exploration step */`
    Query predictor i_t;
    set $\hat{f}_t(i_t) := f_t(i_t)/(2D)$ and $\hat{f}_t(j) = 0\,\forall j \neq i_t$;
    feed $\hat{f}_t$ into $\bar{A}$;
    serve the request at $g_t := \min_x\{d(b_{t-1}, x) + c_t(x)\}$; `/* greedy step */`
    return to $b_t := b_{t-1}$;
  else `/* exploitation step */`
    Feed $\hat{f}_t := 0$ into $\bar{A}$;
    Query predictor a_t chosen by $\bar{A}$ and set $b_t := \varphi_{a_t t}$;

Observation 5.2. Let $X_t = X \cap [t]$ and $I_t = (i_t)_{t \in X_t}$. Since each t was added to X independently at random and i_t was also chosen independently, we have

$$\mathbb{E}[\hat{f}_t | X_{t-1}, I_{t-1}] = \mathbb{E}[\hat{f}_t] = \frac{\gamma}{2D\ell} f_t.$$

We choose $\bar{A}$ to be the algorithm SHARE by Herbster & Warmuth (1998), which has the following advantages over ODDEXPONENT. First, it does not require splitting cost functions as far as they are bounded by 1. Splitting the cost functions is problematic in the Bandit access model, since we are allowed to query only one algorithm per time step. Second, it chooses its state without lookahead, i.e., its state at time t depends only on cost functions $c_1, \dots, c_{t-1}$. See Appendix A for a description of both algorithms.

Proposition 5.3 (Blum & Burch (2000)). *Given $r > 0$, configure* SHARE *with $\alpha = 1/(2r+1)$ and $\beta = \max\{1/2, 1-\gamma\}$, where $\gamma = \frac{1}{r}\ln(\ell/\alpha)$. Then, in the uniform metric space on ℓ points,* SHARE *has r-unfair competitive ratio at most*

$$R^r_\ell := 1 + \frac{8}{r}(\ln \ell + \ln(2r+1)).$$

For $\epsilon > 0$, let $r(\epsilon) = O(\epsilon^{-1}\ln(2+\epsilon^{-1})\ln \ell)$ be such that $R_\ell^{r(\epsilon)} = 1+\epsilon$.

First, we analyze the following variant BANDITCOMBINE′ which queries two predictors during exploration steps. I.e., instead of the greedy step (Lines 8, 9), it makes an additional query to the predictor a_t suggested by $\bar{A}$ and moves to $b'_t := \varphi_{a_t t}$.

Lemma 5.4. *Choose $\bar{A}$ to be an algorithm for MTS on the ℓ-point uniform metric whose ρ-unfair competitive ratio is R with additive term α for instances with bounded cost functions $c_t \leq 1$ and which does not use lookahead, i.e., its state a_t depends only on costs up to time $t-1$. Then the expected cost of* BANDITCOMBINE′ *is at most*

$$R \cdot \mathrm{DYN}_{\rho'} + \frac{2D\ell}{\gamma}\alpha,$$

where $\rho' = \frac{2D\ell\rho}{\gamma}$.

Proof. Let $p_1, \dots p_T \in [0,1]^\ell$ be the probability distributions over the state of $\bar{A}$ at time steps $1,\dots,T$. Note that the expected service cost paid by BANDITCOMBINE′ at time t equals $\langle f_t, p_t\rangle$, the scalar product between the vectors representing the predictor costs and the predictor probabilities. We define d_E as the earth mover's distance between two probability vectors over the uniform metric, which represents the total probability mass that has to be shifted to transform one vector into the other. We may assume that the probability that $\bar{A}$ changes states between times $t-1$ and t equals $d_E(p_{t-1}, p_t)$, as this is the best way to match given probability vectors, which are the core of the algorithm $\bar{A}$. We have

$$\begin{aligned}\mathbb{E}\left[\sum_t \langle \hat{f}_t, p_t\rangle\right] &= \sum_t \mathbb{E}_{X_{t-1}, I_{t-1}}\left[\langle \mathbb{E}[\hat{f}_t | X_{t-1}, I_{t-1}], p_t\rangle\right] \\ &= \frac{\gamma}{2D\ell}\mathbb{E}\left[\sum_t \langle f_t, p_t\rangle\right]\end{aligned}$$

The first equation separates the random events before and after time t, as p_t and p_{t-1} depend solely on X_{t-1} and I_{t-1} while $\hat{f}_t$ is uncorrelated to these events. This allows to use Observation 5.2 to express $\hat{f}_t$ in function of f_t. The last expectation is an upper bound on the expected cost paid by BANDITCOMBINE′, excluding the cost it pays for switching between predictors.

Now, for any instantiation of $\hat{f}$ and for any solution q, we have

$$\begin{aligned}\sum_t (\langle \hat{f}_t, p_t\rangle + d_E(p_{t-1}, p_t)) \leq \\ R\sum_t (\langle \hat{f}_t, q_t\rangle + \rho\, d_E(q_{t-1}, q_t)) + \alpha\end{aligned}$$

by the performance guarantees of $\bar{A}$. Therefore, the total cost of BANDITCOMBINE′ is at most (noting that $\gamma < 2\ell$)

$$\begin{aligned}&\mathbb{E}\left[\sum_t (\langle f_t, p_t\rangle + D\, d_E(p_{t-1}, p_t)\right] \\ &\leq \frac{2D\ell}{\gamma}\mathbb{E}\left[\sum_t (\langle \hat{f}_t, p_t\rangle + d_E(p_{t-1}, p_t))\right] \\ &\leq \frac{2D\ell}{\gamma}\mathbb{E}\left[R\sum_t (\langle \hat{f}_t, q_t\rangle + \rho\, d_E(q_{t-1}, q_t)) + \alpha\right] \\ &= R\sum_t (\langle f_t, q_t\rangle + \frac{2D\ell\rho}{\gamma}\, d_E(q_{t-1}, q_t)) + \frac{2D\ell}{\gamma}\alpha.\end{aligned}$$

Since this is true for any solution q, we get the desired bound. $\square$

Lemma 5.5. *The cost of* BANDITCOMBINE *is at most*

$$(1+6\gamma)\,\mathrm{cost}(\text{BANDITCOMBINE}').$$

The proof of this lemma is deferred to Appendix D.

Proof of Theorem 1.7. We choose $\gamma = \epsilon/6$. By Lemma 5.4, BANDITCOMBINE′ is $(1+\epsilon)$-competitive against $\mathrm{DYN}_{\rho'}$ for $\rho' = \frac{2D\ell}{\gamma}\rho = \frac{2D\ell}{\gamma}r(\epsilon)$. Hence, by Lemma 3.4 it is $(1+\epsilon)^2$-competitive against $\mathrm{DYN}^{\leq m}$ for any $m \leq \frac{\epsilon\gamma}{2\ell r(\epsilon)}\frac{\mathrm{DYN}}{D}$. By Proposition 5.3, the latter quantity is $\Omega\left(\frac{\epsilon^3}{\ell\ln\ell\ln(2+\frac{1}{\epsilon})}\cdot\frac{\mathrm{DYN}}{D}\right)$. Lemma 5.5 implies the theorem. $\square$

References

Almanza, M., Chierichetti, F., Lattanzi, S., Panconesi, A., and Re, G. Online facility location with multiple advice. In *NeurIPS*, pp. 4661–4673, 2021.

Anand, K., Ge, R., Kumar, A., and Panigrahi, D. Online algorithms with multiple predictions. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pp. 582–598. PMLR, 2022.

Antoniadis, A., Coester, C., Eliás, M., Polak, A., and Simon, B. Online metric algorithms with untrusted predictions. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 345–355, 2020.

Antoniadis, A., Coester, C., Eliás, M., Polak, A., and Simon, B. Learning-augmented dynamic power management with multiple states via new ski rental bounds. In *NeurIPS*, 2021.

Bamas, É., Maggiori, A., Rohwedder, L., and Svensson, O. Learning augmented energy minimization via speed scaling. In *NeurIPS*, 2020.

Bansal, N., Coester, C., Kumar, R., Purohit, M., and Vee, E. Learning-augmented weighted paging. In *SODA*, 2022.

Bartal, Y., Blum, A., Burch, C., and Tomkins, A. A polylog(n)-competitive algorithm for metrical task systems. In Leighton, F. T. and Shor, P. W. (eds.), *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pp. 711–719. ACM, 1997. doi: 10.1145/258533.258667. URL https://doi.org/10.1145/258533.258667.

Bhaskara, A., Cutkosky, A., Kumar, R., and Purohit, M. Online linear optimization with many hints. In *NeurIPS*, 2020.

Blum, A. and Burch, C. On-line learning and the metrical task system problem. *Mach. Learn.*, 39(1):35–58, 2000. doi: 10.1023/A:1007621832648.

Borodin, A., Linial, N., and Saks, M. E. An optimal online algorithm for metrical task system. *J. ACM*, 39(4): 745–763, 1992. doi: 10.1145/146585.146588.

Bubeck, S., Cohen, M. B., Lee, J. R., and Lee, Y. T. Metrical task systems on trees via mirror descent and unfair gluing. *SIAM J. Comput.*, 50(3):909–923, 2021. doi: 10.1137/19M1237879. URL https://doi.org/10.1137/19M1237879.

Bubeck, S., Coester, C., and Rabani, Y. The randomized k-server conjecture is false! *CoRR*, abs/2211.05753, 2022a. doi: 10.48550/arXiv.2211.05753.

Bubeck, S., Coester, C., and Rabani, Y. Shortest paths without a map, but with an entropic regularizer. In *FOCS*, pp. 1102–1113. IEEE, 2022b.

Cesa-Bianchi, N. and Lugosi, G. *Prediction, learning, and games*. Cambridge University Press, 2006. ISBN 978-0-521-84108-5. doi: 10.1017/CBO9780511546921. URL https://doi.org/10.1017/CBO9780511546921.

Chrobak, M. and Larmore, L. L. The server problem and on-line games. In *On-Line Algorithms*, volume 7 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pp. 11–64. DIMACS/AMS, 1991.

Chrobak, M., Karloff, H. J., Payne, T. H., and Vishwanathan, S. New results on server problems. *SIAM J. Discret. Math.*, 4(2):172–181, 1991.

Dinitz, M., Im, S., Lavastida, T., Moseley, B., and Vassilvitskii, S. Algorithms with prediction portfolios. In *NeurIPS*, 2022.

Emek, Y., Fraigniaud, P., Korman, A., and Rosén, A. Online computation with advice. In Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S. E., and Thomas, W. (eds.), *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, volume 5555 of *Lecture Notes in Computer Science*, pp. 427–438. Springer, 2009. doi: 10.1007/978-3-642-02927-1_36. URL https://doi.org/10.1007/978-3-642-02927-1_36.

Emek, Y., Kutten, S., and Shi, Y. Online paging with a vanishing regret. In *ITCS*, 2021.

Fiat, A. and Mendel, M. Better algorithms for unfair metrical task systems and applications. *SIAM J. Comput.*, 32(6):1403–1422, 2003. doi: 10.1137/S0097539700376159. URL https://doi.org/10.1137/S0097539700376159.

Fiat, A., Rabani, Y., and Ravid, Y. Competitive k-server algorithms (extended abstract). In *FOCS*, 1990.

Fiat, A., Foster, D. P., Karloff, H. J., Rabani, Y., Ravid, Y., and Vishwanathan, S. Competitive algorithms for layered graph traversal. *SIAM J. Comput.*, 28(2):447–462, 1998.

Gollapudi, S. and Panigrahi, D. Online algorithms for rent-or-buy with expert advice. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2319–2327. PMLR, 2019.

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

Im, S., Kumar, R., Petety, A., and Purohit, M. Parsimonious learning-augmented caching. In *ICML*, 2022.

Lykouris, T. and Vassilvitskii, S. Competitive caching with machine learned advice. *J. ACM*, 68(4):24:1–24:25, 2021.

Manasse, M. S., McGeoch, L. A., and Sleator, D. D. Competitive algorithms for server problems. *J. Algorithms*, 11 (2):208–230, 1990.

Purohit, M., Svitkina, Z., and Kumar, R. Improving online algorithms via ML predictions. In *NeurIPS*, 2018.

Rohatgi, D. Near-optimal bounds for online caching with machine learned advice. In *SODA*, 2020.

Slivkins, A. Introduction to multi-armed bandits. *CoRR*, abs/1904.07272, 2019. URL http://arxiv.org/abs/1904.07272.

Wang, S., Li, J., and Wang, S. Online algorithms for multi-shop ski rental with machine learned advice. In *NeurIPS*, 2020.

Wei, A. Better and simpler learning-augmented online caching. In *APPROX/RANDOM*, 2020.

A. Algorithms for unfair MTS

A.1. Odd Exponent

There is a randomized algorithm ODDEXPONENT based on work functions proposed by (Bartal et al., 1997).

Choose an odd number a close to $\ln \ell$ as a parameter. State j is chosen with probability

$$p_j := \frac{1}{\ell} + \frac{1}{\ell}\sum_{i=1}^{\ell}(w_t(i) - w_t(j))^a,$$

where $w_t(i)$ is the work function of state i at time t:

$$w_t(i) = \min\{d(i, x_t) + \sum_{j=1}^{t}(c_j(x_j) + r \cdot d(x_{j-1}, x_j)) \mid x_j \in M\}.$$

In order to be well defined (e.g., for all probabilities to be non-negative), the input sequence $c_1, \dots, c_T$ needs to satisfy the following properties:

- Each c_t has only one non-zero coordinate
- If c_t with non-zero value $c_t(i)$ would make ODDEXPONENT remove all probability mass from state i, we assume that $c_t(i)$ is the smallest such value.
- If ODDEXPONENT already has 0 probability mass at state i, no cost function with $c_t(i) > 0$ arrive.

These properties can be assumed without loss of generality, since they can be achieved by splitting each cost function into several smaller ones and omitting those which do not imply any cost on the algorithm (this omission does not increase the cost of the offline optimum either). We refer to Section 4.4.1 in (Blum & Burch, 2000) for more details on how to implement this algorithm in the general MTS setting.

A.2. SHARE

SHARE is an algorithm for tracking the best expert regime in Online Learning proposed by Herbster & Warmuth (1998). It requires two parameters: the sharing parameter $\alpha \in [0, 1/2]$ and $\beta \in [0, 1]$ (logarithm of the learning rate).

We can apply it to unfair MTS in uniform metric space of size ℓ with cost functions bounded by 1 as follows. It starts with weights $w_0(1) = \cdots w_0(\ell) = 1$ and uniform probability distribution over the states, i.e., $p_0(i) = w_0(i)/\sum_{j=1}^{\ell} w_0(j)$. At time t, when its probability distribution over states is p_t, it incurs cost $\langle p_t, c_t \rangle$ and updates the weights and its probability distribution for the next time step:

$$w_{t+1}(i) := w_t(i) \cdot \beta^{c_t(i)} + \alpha\Delta/\ell$$

$$p_{t+1}(i) := w_{t+1}(i)/\sum_{j=1}^{\ell} w_{t+1}(j),$$

where $\Delta = \sum_{i=1}^{\ell}(w_t(i) - w_t(i)\beta^{c_t(i)})$. This way, its distribution p_t depends only on $c_1, \dots, c_{t-1}$ and proposes only one distribution p_t at each time step. Proposition 5.3 by Blum & Burch (2000) states the performance guarantee of this algorithm for unfair MTS. Note that this algorithm can be easily adapted to unbounded cost functions: we split each cost function into several smaller cost functions bounded by 1. Due to this splitting, however, it may move several times during each time step.

B. Omitted proof of Theorem 1.3

Proof of Theorem 1.3. By the result of Bubeck et al. (2022a) that any (randomized) algorithm for MSS with $w = \ell$ is $\Omega(\ell^2)$-competitive, and since k-server is an MTS, it is sufficient for proving the lemma to show that MSS with $w = \ell$ is a special case of the k-server problem.

For k-server, consider an arbitrary input instance I to MSS on a metric $\mathcal{M}$ on n many points where the server initially is at a point $p_0 \in \mathcal{M}$. We can initialize a k-server instance I' with $k = n - 1$ on the same metric, by placing one server on each point other than p_0. Fix a learning-augmented algorithm A' for k-server with full access to ℓ predictors. We define an algorithm A for MSS via A' as follows. Whenever a request to a set W_i arrives in I on round i we in I' repeatedly issue requests to the currently empty point (starting with p_i) and let A' serve these. For each of these requests $p_i^1 = p_i, p_i^2, \dots$ let the ℓ many predictors collectively predict the points in W_i. The sequence ends with the first request to a point $p_i^z = q \in W_i$, at which point A serves the original request W_i in I with point q. Note that z need not be finite, but if it is, then the single server in I is located on $p_{i+1} = q$ and the only point on which A' has no server is also p_{i+1}.

Furthermore the cost that A' incurs on this set of requests is $dist(p_i = p_i^1, p_i^2) + dist(p_i^2, p_i^3) + \dots + dist(p_i^{z-1}, p_i^z = q)$ which for finite z is at least $dist(p_i, p_{i+1})$. The latter term is exactly the cost of A for the current request. In turn, the existence of an $o(\ell^2)$-competitive against DYN algorithm A' for k-server with ℓ predictors would contradict the result of Bubeck et al. (2022a). □

C. Lower bound for k-server against $\widetilde{\mathrm{DYN}}$

In their recent work, Anand et al. (2022) expressed the belief that their framework for multiple predictions can be applied to problems other than set-cover, (weighted) caching and facility location. In particular "it would be interesting to consider the k-server problem with multiple suggestions in each step specifying the server that should serve the new request". For the benchmark DYN, we gave a tight answer of $\Theta(\ell^2)$ in Section 2. But also for the benchmark $\widetilde{\mathrm{DYN}}$, we show that there exist instances on which such predictors are not beneficial.

Proof of Theorem 1.1. Consider the line metric with $k+1$ distinct points indexed from left to right as $p_1, p_2, \dots p_{k+1}$. As mentioned in the introduction, we can restrict to *lazy* algorithms. Furthermore, we can assume without loss of generality that for any algorithm, two servers never reside at the same point simultaneously.

It is known (Manasse et al., 1990) (resp. (Bubeck et al., 2022a)) that any deterministic (resp. randomized) online algorithm has competitive ratio at least k (resp. $\Omega(\log k)$) on any metric space of at least $k+1$ points. Let the set of servers be indexed $s_1, \dots s_k$ from left to right in their initial configuration. In order to prove the theorem, it is sufficient to show that there exists an optimal solution OPT on which every request to a point p_i is served by servers s_{i-1} or s_i. The result then follows by having the two predictors produce suggestions s_{i-1} and s_i respectively (for the border cases when $i = 1$ or $i = k+1$ we have both predictors suggest s_1 or s_k respectively) whenever point p_i is requested. This implies that OPT is an algorithm that serves each request p_i using a server named by one of the predictors in that round and thus by definition cannot have cost lower than $\widetilde{\mathrm{DYN}}$.

For the sake of contradiction assume that the claim is wrong, that is, there exists some optimal algorithm OPT which serves some request p_i in round r with a server s_j such that $j < i-1$ or $j > i$. In case there are more such algorithms, let OPT be one maximizing r. We assume $j < i-1$ as the other case is symmetrical. We modify OPT to obtain an algorithm OPT$'$ as follows. The rounds up to (excluding) r are served identically to OPT. The request to p_i in round r is served by server s_{i-1} (which currently resides at p_{i-1}). At the same time, the server s_j moves to p_{i-1}, so that servers s_{i-1} and s_j are swapped compared to the current state of OPT. In later rounds, OPT$'$ imitates OPT but exchanging the roles of servers s_{i-1} and s_j. This gives an algorithm with the same cost as OPT thus contradicting the definition of r.

□

If the learning-augmented algorithm is forced to follow a predictor's suggestion in each step, then the above proof extends to arbitrary metric spaces by fixing $k+1$ points $p_1, \dots, p_{k+1}$ to be used for the lower bound instance, and using two predictors that keep their ith server always at one of the two points p_i and p_{i+1} so that the set of usable edges constitutes a path.

D. Omitted proof of Lemma 5.5

Proof. Note that states of BanditCombine and BanditCombine$'$ are the same during exploitation steps, i.e., $b_t = b'_t$ for all exploitation steps. At time step t, the cost paid by BanditCombine$'$ is $C'_t = d(b'_{t-1}, b'_t) + c_t(b'_t)$ and its total cost is $\sum_{t=1}^{T} C'_t$.

To bound the cost paid by BanditCombine we define $C_t := d(b'_{t-1}, b_t) + c_t(b_t)$ if t is an exploitation step, note that

$b_t = b'_t$ in such case. For exploration steps, we define

$$C_t := d(b'_{t-1}, g_t) + c_t(g_t) + d(g_t, b_t) + d(b_t, b'_t).$$

The last term is to simplify the analysis: at step $t+1$, BANDITCOMBINE moves by a distance $d(b_t, b_{t+1}) \le d(b_t, b'_t) + d(b'_t, b_{t+1})$ and we split this cost counting the first part to C_t and the second one to C_{t+1}. The total cost of BANDITCOMBINE is then at most $\sum_{t=1}^{T} C_t$ and we have $C_t = C'_t$ for every exploitation step t.

Consider an exploration step t which is the ith consecutive exploration step, i.e. step $a = t - i$ is an exploitation step (or $a = 0$) and all steps from $a+1$ until t are exploration. Observe that BANDITCOMBINE does not change b_t during exploration steps and we have $b_t = b_a = b'_a$. We can bound C_t as follows. We have

$$\begin{aligned} d(b_t, b'_t) = d(b_a, b'_t) &\le C'_{t-i+1} + \cdots + C'_t \\ d(b'_{t-1}, g_t) &\le d(b'_{t-1}, b_t) + d(b_t, g_t) \\ d(b_t, g_t) + c_t(g_t) &\le d(b_t, b'_t) + c_t(b'_t) \end{aligned}$$

The first inequality holds because BANDITCOMBINE$'$ needs to move from $b_t = b_{t-i}$ to b'_t. The second one follows from the triangle inequality and the last one comes from the definition of g_t. In total, we have

$$C_t \le \big[d(b'_{t-1}, b_t)\big] + \big[d(b_t, g_t) + c_t(g_t) + d(g_t, b_t)\big] + \big[d(b_t, b'_t)\big].$$

The first and the third brackets are bounded by $\sum_{j=a+1}^{t} C'_j$, because $b_t = b'_a$ and BANDITCOMBINE$'$ moves from b_t to b'_{t-1} and b'_t during that time. By the choice of g_t, the second bracket is at most $2(d(b_t, b'_t) + c_t(b'_t)) \le 2\sum_{j=a+1}^{t} C'_j$.

Since the probability of t being the ith exploration step in a row is at most $(1-\gamma)\,\gamma^i$, the expected cost of BANDITCOMBINE is at most

$$\begin{aligned} &(1-\gamma)\sum_t C'_t + \sum_{i=1}^{T}(1-\gamma)\gamma^i \cdot \sum_t 4 \sum_{j=t-i}^{t} C'_j \\ &\le (1-\gamma)\sum_t C'_t + \sum_{i=1}^{T}\gamma^i 4i \sum_t C'_t \end{aligned}$$

which is at most $(1+6\gamma)\sum_t C'_t$ for $\gamma \le 1/4$. □