Online Algorithms with Multiple Predictions

Keerti Anand [1] **Rong Ge** [1] **Amit Kumar** [2] **Debmalya Panigrahi** [1]

Abstract

This paper studies online algorithms augmented with *multiple* machine-learned predictions. We give a generic algorithmic framework for online covering problems with multiple predictions that obtains an online solution that is competitive against the performance of the *best* solution obtained from the predictions. Our algorithm incorporates the use of predictions in the classic potential-based analysis of online algorithms. We apply our algorithmic framework to solve classical problems such as online set cover, (weighted) caching, and online facility location in the multiple predictions setting.

1. Introduction

In many real world computational tasks, parts of the input are not known in advance and are revealed piecemeal over time. However, the algorithm is constrained to take decisions before the entire input is revealed, thereby optimizing for an unknown future. For instance, when an online retailer has to decide the locations of warehouses to serve clients, it does so without precisely knowing how the clientele will grow over time. Similarly, in an operating system, the cache scheduler has to decide which pages to evict from the cache without knowing future requests for page access. These kinds of scenarios are traditionally captured by the field of *online algorithms*, where the algorithm makes irrevocable decisions without knowing the future. The performance of an online algorithm is measured by its *competitive ratio* which is defined as the maximum ratio across all inputs between the cost of the online algorithm and that of an optimal solution (see, e.g., (Borodin & El-Yaniv, 1998)). While this is a robust guarantee that holds for *all* inputs, the robustness comes at the cost of making online algorithms excessively cautious thereby resulting in strong lower bounds and also affecting their real world performance.

To overcome the pessimistic behavior of online algorithms, there has been a growing trend in recent years to incorporate machine-learned predictions about the future. This exploits the fact that in many real world settings, modern ML methods can predict future behavior to a high degree of accuracy. Formalized by Lykouris and Vassilvitksii (Lykouris & Vassilvitskii, 2018; 2021) for the caching problem, the online algorithms with prediction framework allows online algorithms to access predicted future input values, but does not give any guarantee on the accuracy of such predictions. (This reflects the fact that ML predictions, say generated by a neural network, are usually without worst-case guarantees, and can occasionally be completely wrong.) The goal is to design online algorithms whose competitive ratio gracefully interpolates between offline algorithms if the predictions are accurate – a property called *consistency* – and online algorithms irrespective of predictions – a property called *robustness* (these terms were coined by Kumar, Purohit, and Svitkina (Kumar et al., 2018)).

Online algorithms with predictions have been extensively studied in the last few years for a broad range of problems such as variants of ski rental (Kumar et al., 2018; Khanafer et al., 2013; Gollapudi & Panigrahi, 2019; Wei & Zhang, 2020; Anand et al., 2020; Wang et al., 2020), set cover (Bamas et al., 2020b), scheduling (Kumar et al., 2018; Wei & Zhang, 2020; Bamas et al., 2020a; Lattanzi et al., 2020; Mitzenmacher, 2020; Lee et al., 2021; Azar et al., 2021), caching (Lykouris & Vassilvitskii, 2018; Wei, 2020; Jiang et al., 2020; Bansal et al., 2020), matching and secretary problems (Lavastida et al., 2021; Dütting et al., 2021; Antoniadis et al., 2020b; Jiang et al., 2021b), metric optimization (Antoniadis et al., 2020a; Azar et al., 2022; Fotakis et al., 2021; Jiang et al., 2021a; Almanza et al., 2021), data structures (Mitzenmacher, 2018), statistical estimation (Hsu et al., 2019; Indyk et al., 2019; Eden et al., 2021), online search (Anand et al., 2021), and so on.

In this paper, we focus on online algorithms with *multiple* machine-learned predictions. In many situations, different ML models and techniques end up with distinct predictions about the future, and the online algorithm has to decide which prediction to use among them. Indeed, this is also true of human experts providing inputs about expectations

[1]Department of Computer Science, Duke University, Durham, NC, USA. [2]Department of Computer Science and Engineering, IIT Delhi, New Delhi, India.. Correspondence to: Keerti Anand <kanand@cs.duke.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

of the future, or other statistical tools for predictions such as surveys, polls, etc. Online algorithms with multiple predictions were introduced by Gollapudi and Panigrahi (Gollapudi & Panigrahi, 2019) for the ski rental problem, and has since been studied for multi-shop ski rental (Wang et al., 2020) and facility location (Almanza et al., 2021). Furthermore, (Bhaskara et al., 2020) considers multiple hints for regret minimization in Online Linear Optimization. In our current paper, instead of focusing on a single problem, we extend the powerful paradigm of *online covering problems* to incorporate multiple predictions. As a consequence, we obtain online algorithms with multiple predictions for a broad range of classical problems such as *set cover*, *caching*, and *facility location* as corollaries of the general technique that we develop in this paper.

The Online Covering Framework. Online covering is a powerful framework for capturing a broad range of problems in combinatorial optimization. In each online step, a new linear constraint $\mathbf{a} \cdot \mathbf{x} \geq b$ is presented to the algorithm, where $\mathbf{x}$ is the vector of variables, $\mathbf{a}$ is a vector of non-negative coefficients, and b is a scalar. The algorithm needs to satisfy the new constraint, and is only allowed to increase the values of the variables to do so. The goal is to minimize an objective function $\mathbf{c} \cdot \mathbf{x}$, where $\mathbf{c}$ is the vector of costs that is known offline. This formulation captures a broad variety of problems including set cover, (weighted) caching, revenue maximization, network design, ski rental, TCP acknowledgment, etc. Alon *et al.* (Alon et al., 2009) proposed a multiplicative weights update (MWU) technique for this problem and used it to solve the online set cover problem. This was quickly adapted to other online covering problems including weighted caching (Bansal et al., 2007), network design (Alon et al., 2006), allocation problems for revenue maximization (Buchbinder et al., 2007), etc. (The reader is referred to the survey (Buchbinder & Naor, 2009b) for more examples.) All these algorithms share a generic method for obtaining a fractional solution to the online covering problem, which was formalized by Buchbinder and Naor (Buchbinder & Naor, 2009a) and later further refined by Gupta and Nagarajan (Gupta & Nagarajan, 2014). Since then, the online covering problem has been generalized to many settings such as convex (non-linear) objectives (Azar et al., 2016) and mixed covering and packing problems (Azar et al., 2013).

Comparison with Prior Work on Online Covering with ML Prediction. Bamas, Maggiori, and Svensson (Bamas et al., 2020b) were the first to consider the online covering framework in the context of ML predictions. In a beautiful work, they gave the first general-purpose tool for online algorithms with predictions, and showed that this can be used to solve several classical problems like set cover and dynamic TCP acknowledgment. In their setting, a solution is presented as advice to the online algorithm at the outset, and the algorithm incorporates this suggestion in its online decision making.

In our current paper, we give a general scheme for the online covering framework with *multiple* predictions. In particular:

– Since we are in the multiple predictions setting, we allow $k > 1$ suggestions instead of just a single suggestion, and benchmark our algorithm's performance against the best suggested solution. (Of course, the best suggestion is not known to the algorithm.)

– In contrast to Bamas *et al.* (Bamas et al., 2020b), we do not make the assumption that the entire suggested solution is given up front. Instead, in each online step, each of the k suggestions gives a feasible way of satisfying the new constraint. Note that this is more general than giving the suggested solution(s) up front, since the entire solution(s) can be presented in each online step as a feasible way of satisfying the new constraint.

– In terms of the analysis, we extend the potential method from online algorithms (in contrast, Bamas *et al.* (Bamas et al., 2020b) use the primal dual framework). The potential method has been used recently for many classic problems in online algorithms such as weighted paging (Bansal et al., 2010), k-server (Buchbinder et al., 2019), metric allocation (Bansal & Coester, 2021), online set cover (Buchbinder et al., 2019), etc. In fact, it can also be used to reprove the main results of Bamas *et al.* (Bamas et al., 2020b) in the single prediction setting. In this paper, we extend this powerful tool to incorporate multiple ML predictions.

– Finally, we show that our techniques extend to a generalization of the online covering framework to include *box*-type constraints. This extension allows the framework to handle more problems such as online facility location that are not directly captured by the online covering framework.

Comparison with Online Learning. The reader will notice the similarity of our problem to the classical experts' framework from online learning (see, e.g., the survey (Shalev-Shwartz, 2012)). In the experts' framework, each of k experts provides a suggestion in each online step, and the algorithm has to choose (play) one of these k options. After the algorithm has made its choice, the cost (loss) of each suggestion is revealed before the next online step. The goal of the algorithm is to be competitive with the *best expert* in terms of total loss. In contrast,

– Since we are solving a combinatorial problem, the (incremental) cost of any given step for an expert or the algorithm depends on their pre-existing solution from previous steps (therefore, in particular, even after following an expert's choice, the algorithm might suffer a larger incremental cost than the expert). This is unlike online learning where the cost in a particular step is independent of previous choices.

- In online learning, the algorithm is benchmarked against the best *static* expert in hindsight, i.e., the best solution whose choices match that of the same expert across all the steps. Indeed, it can be easily shown that no algorithm can be competitive against a *dynamic* expert, namely a solution that chooses the best suggestion in each online step even if those choices come from different experts. Observe that such a dynamic expert can in general perform much better than each of the suggestions, e.g., when the suggestions differ from each other but at each time, at least one of them suggests a good solution. But, in our problem, since the choices made by experts correspond to solutions of a combinatorial problem, we can actually show that our algorithm is competitive even against a dynamic expert. Namely, the k suggestions in every step are not indexed by specific experts, and the algorithm is competitive against any composite solution that chooses any one of the k suggestions in each step.
- In online learning, the goal is to obtain *regret* bounds that show that the online algorithm approaches the best (static) expert's performance up to additive terms. Such additive guarantees are easily ruled out for our problem, even for a static expert. As is typically the case in online algorithms, our performance guarantees are in the form of (multiplicative) competitive ratios rather than (additive) regret bounds.

Our Contributions. Our first contribution is to formalize the online covering problem with multiple predictions (Section 2). Recall that in each online step, along with a new constraint, the algorithm receives k feasible suggestions for satisfying the constraint. Using these suggestions, we design an algorithm for obtaining a fractional solution to the online covering problem–that we call the OCP algorithm (Section 3). To compare this algorithm to the best suggested solution, we define a benchmark DYNAMIC that captures the minimum cost (fractional) solution that is consistent with at least one of the suggestions in each online step.

- Our main technical result shows that the cost of the solution produced by the OCP algorithm is at most $O(\log k)$ times that of the DYNAMIC solution.

It is noteworthy that unlike in the classical online covering problem (without predictions), the competitive ratio is independent of the problem size, and only depends on the number of suggestions k. As the number of suggestions increases, the competitive ratio degrades because the suggestions have higher entropy (i.e., are less specific). As two extreme examples, consider $k = 1$, in which case it is trivial for an algorithm to exactly match the DYNAMIC benchmark simply by following the suggestion in each step. In contrast, when k is very large, the set of suggested solutions can essentially include all possible solutions, and therefore, the suggestions are useless.

The analysis of the OCP algorithm makes careful use of potential functions that might be of independent interest. But, while the analysis of the OCP algorithm is somewhat intricate, we note that the algorithm itself is extremely simple.

- We show that the competitive ratio of $O(\log k)$ obtained by the OCP algorithm is tight. We give a lower bound of $\Omega(\log k)$ by only using binary (0/1) coefficients and unit cost for each variable, which implies that the lower bound holds even for the special case of the unweighted set cover problem.
- Using standard techniques, we observe that the OCP algorithm can be *robustified*, i.e., along with being $O(\log k)$-competitive against the best suggested solution, the algorithm can be made $O(\alpha)$-competitive against the optimal solution where α is the competitive ratio of the best online algorithm (without predictions).

We then use the OCP algorithm to solve two classic problems–online **set cover** (Section 4) and **caching** (Section 5)–in the multiple predictions setting.

- We generalize the online covering framework by introducing *box*-type constraints (Section 6). We show that our techniques and results from online covering extend to this more general setting.

We then use this more general formulation for solving the classical online **facility location** problem (Section 7).

2. The Online Covering Framework

2.1. Problem Statement

We define the *online covering problem* (OCP) as follows. There are n non-negative variables $\{x_i : i \in [n]\}$ where each $x_i \in [0, 1]$. Initially, $x_i = 0$ for all $i \in [n]$. A linear objective function $c(x) := \sum_{i=1}^{n} c_i x_i$ is also given offline. In each online step, a new *covering* constraint is presented, the j-th constraint being given by $\sum_i a_{ij} x_i \geq 1$ where $a_{ij} \geq 0$ for all $i \in [n]$.[1] The algorithm is only allowed to

[1] A more general definition allows constraints of the form $\sum_{i=1}^{n} a_{ij} x_i \geq b_j$ for any $b_j > 0$, but restricting b_j to 1 is without loss of generality since we can divide throughout by b_j without changing the constraint.

increase the values of the variables, and has to satisfy the new constraint when it is presented. (We denote the total number of constraints by m.) The goal is to minimize the objective function $c(x)$. We write this succinctly below:

$$\min_{x_i \in [0,1]: i \in [n]} \left\{ \sum_{i=1}^{n} c_i x_i : \sum_{i=1}^{n} a_{ij} x_i \geq 1 \; \forall j \in [m] \right\}.$$

This framework captures a large class of algorithmic problems such as (fractional) set cover, caching, etc. that have been extensively studied in the online algorithms literature. Our goal will be to obtain a generic algorithm for OCP with multiple suggestions. When the j-th constraint is presented online, the algorithm also receives k suggestions of how the constraint can be satisfied. We denote the s-th suggestion for the j-th constraint by variables $x_i(j,s)$; they satisfy $\sum_{i=1}^{n} a_{ij} x_i(j,s) \geq 1$, i.e., all suggestions are feasible.

To formally define the *best* suggestion, we say that a solution $\{x_i : i \in [n]\}$ is *supported* by the suggestions $\{x_i(j) : i \in [n], j \in [m]\}$ if $x_i \geq x_i(j)$ for all $j \in [m]$. Using this definition, we consider below two natural notions of the best suggestion that we respectively call the *experts setting* and the *multiple predictions setting*.

The Experts Setting. In the experts setting, there are k experts, and the s-th suggestion for each constraint comes from the same fixed expert $s \in [k]$ (say some fixed ML algorithm or a human expert). The online algorithm is required to be competitive with the *best* among these k experts[2]. To formalize this, we define the benchmark:

$$\text{STATIC} = \min_{s \in [k]} \sum_{i=1}^{n} c_i \cdot \max_{j \in [m]} x_i(j,s).$$

Note that $\{\max_{j \in [m]} x_i(j,s) : i \in [n]\}$ is the *minimal* solution that is supported by the suggestions of expert s; hence, we define the cost of the solution proposed by expert s to be the cost of this solution.

The Multiple Predictions Setting. In the multiple predictions setting, we view the set of k suggestions in each step as a bag of k predictions (without indexing them specifically to individual predictors or experts) and the goal is to obtain a solution that can be benchmarked against the best of these suggestions in each step. Formally, our benchmark is the minimum-cost solution that is supported by at least one suggestion in each online step:

$$\text{DYNAMIC} = \min_{\hat{\mathbf{x}} \in \hat{X}} \sum_{i=1}^{n} c_i \cdot \hat{x}_i, \text{ where}$$

$$\hat{X} = \{\hat{\mathbf{x}} : \forall i \in [n], \forall j \in [m], \exists s \in [k], \hat{x}_i \geq x_i(j,s)\}.$$

Note that every solution that is supported in the experts setting is also supported in the multiple predictions setting. This implies that STATIC $\geq$ DYNAMIC, and therefore, the competitive ratios that we obtain in the multiple predictions setting also hold in the experts setting. Conversely, the lower bounds on the competitive ratio that we obtain in the experts setting also hold in the multiple predictions setting.

[2]This is similar to the experts model in online learning, hence the name.

2.2. Our Results

We obtain an algorithm for OCP with the following guarantee in the multiple predictions setting (and therefore also in the experts setting by the discussion above):

Theorem 2.1. *There is an algorithm for the online covering problem with k suggestions that has a competitive ratio of $O(\log k)$, even against the* DYNAMIC *benchmark.*

Note that this competitive ratio is independent of the size of the problem instance, and only depends on the number of suggestions. In contrast, in the classical online setting, the competitive ratio (necessarily) depends on the size of the problem instance.

Next, we show that the competitive ratio in Theorem 2.1 is tight by showing a matching lower bound. This lower bound holds even in the experts setting (hence, by the discussion above, it automatically extends to the multiple predictions setting):

Theorem 2.2. *The competitive ratio of any algorithm for the online covering problem with k suggestions is $\Omega(\log k)$, even against the* STATIC *benchmark.*

We noted earlier that it is desirable for online algorithms to have *robustness* guarantees, i.e., that the algorithm does not fare much worse than the best online algorithm (without predictions) even if the predictions are completely inaccurate. Our next result is the robust version of Theorem 2.1:

Theorem 2.3. *Suppose a class of online covering problems have an online algorithm (without predictions) whose competitive ratio is α. Then, there is an algorithm for this class of online covering problems with k suggestions that produces an online solution whose cost is at most $O(\min\{\log k \cdot \text{DYNAMIC}, \alpha \cdot \text{OPT}\})$.*

We will prove Theorem 2.1 in the next section. The proofs of Theorem 2.2 and Theorem 2.3 are given in Section 3.2 and Section 3.1 respectively. Subsequently, we apply the algorithmic framework developed in Theorem 2.1 to obtain tight competitive ratios for specific instantiations of OCP, namely the set cover problem (Section 4) and the caching problem (Section 5). Finally, we extend our OCP result to include box-type constraints (Section 6) and apply it to the online facility location problem (Section 7).

3. Online Covering Algorithm

Recall that in the online covering problem, the new constraint that arrives in the j-th online step is $\sum_{i=1}^{n} a_{ij}x_i \geq 1$ and the algorithm receives k suggestions where the s-th suggestion is denoted $x_i(j,s)$. If the current solution of the algorithm given by the variables x_i is feasible, i.e., $\sum_{i=1}^{n} a_{ij}x_i \geq 1$, then the algorithm does not need to do anything. Otherwise, the algorithm needs to increase these variables until they satisfy the constraint. Next, we describe the rules governing the increase of variables.

Intuitively, the rate of the increase of a variable x_i should depend on three things. First, it should depend on the cost of this variable in the objective, namely the value of c_i; the higher the cost, the slower we should increase this variable. Second, it should depend on the contribution of variable x_i in satisfying the new constraint, namely the value of a_{ij}; the higher this coefficient, the faster should we increase the variable. Finally, the third factor is how *strongly* x_i has been suggested. To encode this mathematically, we first make the assumption that every suggestion is *tight*, i.e.,

$$\sum_{i=1}^{n} a_{ij}x_i(j,s) = 1 \text{ for every suggestion } s \in [k]. \qquad (1)$$

This assumption is without loss of generality because, if not, we can decrease the variables $x_i(j,s)$ in an arbitrary manner until the constraint becomes tight. (Note that this change can only decrease the cost of the benchmark solutions DYNAMIC and STATIC; hence, any competitive ratio bounds obtained after this transformation also hold for the original set of suggestions.)

Having made all the suggestions tight, we now encode how strongly a variable has been suggested by using its *average* suggested value $\frac{1}{k} \cdot \sum_{s=1}^{k} x_i(j,s)$. Our algorithm (see Algorithm 1) increases all variables x_i satisfying $x_i < \frac{1}{2}$ simultaneously at rates governed by these parameters; precisely, we use

$$\frac{dx_i}{dt} = \frac{a_{ij}}{c_i}\left(x_i + \delta \cdot x_{ij}\right), \text{ where } \delta = \frac{1}{k}, x_{ij} = \sum_{s=1}^{k} x_i(j,s).$$

The algorithm continues to increase the variables until $\sum_{i=1}^{n} a_{ij}x_i \geq \frac{1}{2}$; along the way, any variable x_i that reaches $\frac{1}{2}$ is dropped from the set of increasing variables. To satisfy the j-th constraint, we note that the variables $2x_i$ are feasible for the constraint. (Note that since all variables $x_i \leq \frac{1}{2}$ before the scaling, every variable can be doubled without violating $x_i \leq 1$.) Since this last step of multiplying every variable by 2 only increases the cost of the algorithm by a factor of 2, we ignore this last scaling step in the rest of the analysis.

Algorithm 1 Online Covering Algorithm

Offline: All variables x_i are initialized to 0.
Online: On arrival of the j-th constraint:
  while $\sum_{i=1}^{n} a_{ij}x_i < \frac{1}{2}$,
    for $i \in [n]$
      if $x_i < \frac{1}{2}$, increase x_i by $\frac{dx_i}{dt} = \frac{a_{ij}}{c_i}(x_i + \delta \cdot x_{ij})$,
        where $\delta = \frac{1}{k}$ and $x_{ij} = \sum_{s=1}^{k} x_i(j,s)$.

Before analyzing the algorithm, we note that although we described it using a continuous process driven by a differential equation, the algorithm can be easily discretized and made to run in polynomial time where in each discrete step, some variable x_i reaches $\frac{1}{2}$ (and therefore, x_i cannot increase any further) or $\sum_{i=1}^{n} a_{ij}x_i$ reaches $\frac{1}{2}$ (and therefore, the algorithm ends for the current online step). In this section, we will analyze the continuous algorithm rather than the equivalent discrete algorithm for notational simplicity.

Next, we show that the algorithm is valid, i.e., that there is always a variable x_i that can be increased inside the **while** loop. If not, then we have $\sum_{i=1}^{n} a_{ij}x_i < \frac{1}{2}$ but $x_i \geq \frac{1}{2}$ for all variables x_i, $i \in [n]$. This implies that $\sum_{i=1}^{n} a_{ij} < 1$, which is a contradiction because the constraint $\sum_{i=1}^{n} a_{ij}x_i \geq 1$ is then unsatisfiable by any setting of variables $x_i \leq 1$. (In particular, this would mean that there cannot be any feasible suggestion for this constraint.)

Now, we are ready to bound the competitive ratio of Algorithm 1 with respect to the DYNAMIC benchmark. First, we bound the rate of increase of algorithm's cost:

Lemma 3.1. *The rate of increase of cost in Algorithm 1 is at most $\frac{3}{2}$.*

Proof. The rate of increase of cost is given by:

$$\begin{aligned}\sum_{i=1}^{n} c_i \cdot \frac{dx_i}{dt} &= \sum_{i=1}^{n} a_{ij}\left(x_i + \delta \cdot x_{ij}\right) \\ &= \sum_{i=1}^{n} a_{ij}x_i + \frac{1}{k} \cdot \sum_{i=1}^{n}\sum_{s=1}^{k} a_{ij}x_i(j,s) \\ &< \frac{1}{2} + \frac{1}{k} \cdot \sum_{s=1}^{k}\left(\sum_{i=1}^{n} a_{ij}x_i(j,s)\right) = \frac{3}{2},\end{aligned}$$

where we used $\sum_{i=1}^{n} a_{ij}x_i < \frac{1}{2}$ from the condition on the **while** loop, and $\sum_{i=1}^{n} a_{ij}x_i(j,s) = 1$ for all $s \in [k]$ from Equation (1). ☐

We now define a carefully crafted non-negative potential function ϕ. We will show that the potential decreases at constant rate when Algorithm 1 increases the variables x_i (Lemma 3.4). By Lemma 3.1, this implies that the potential can pay for the cost of Algorithm 1 up to a constant. We will also show that the potential ϕ is at most $O(\log k)$ times the DYNAMIC benchmark (Lemma 3.3). Combined, these yield Theorem 2.1.

Let x_i^{DYN} denote the value of variable x_i in the DYNAMIC benchmark. The potential function for a variable x_i is then defined as follows:

$$\phi_i = c_i \cdot x_i^{\text{DYN}} \cdot \ln \frac{(1+\delta)x_i^{\text{DYN}}}{x_i + \delta x_i^{\text{DYN}}}, \text{ where } \delta = \frac{1}{k}.$$

and the overall potential is:

$$\phi = \sum_{i : x_i^{\text{DYN}} \geq x_i} \phi_i.$$

The intuition behind only including those variables that have $x_i^{\text{DYN}} \geq x_i$ in the potential function is that the potential stores the excess cost paid by the DYNAMIC benchmark for these variables so that it can be used later to pay for increase in the algorithm's variables.

First, we verify that the potential function is always non-negative.

Lemma 3.2. *For any values x_i, x_i^{DYN} of the variables, the potential function ϕ is non-negative.*

Proof. Note that ϕ only includes variables x_i such that $x_i^{\text{DYN}} \geq x_i$. For such variables,

$$\phi_i = c_i \cdot x_i^{\text{DYN}} \cdot \ln \frac{(1+\delta)x_i^{\text{DYN}}}{x_i + \delta x_i^{\text{DYN}}} = c_i \cdot x_i^{\text{DYN}} \cdot \ln \frac{1+\delta}{\frac{x_i}{x_i^{\text{DYN}}} + \delta} \geq 0.$$

□

Next, we bound the potential as a function of the variables x_i^{DYN} in the DYNAMIC benchmark:

Lemma 3.3. *The potential ϕ_i for variable x_i is at most $c_i x_i^{\text{DYN}} \cdot \ln\left(1 + \frac{1}{\delta}\right) = c_i x_i^{\text{DYN}} \cdot O(\log k)$. As a consequence, the overall potential $\phi \leq O(\log k) \cdot \sum_{i=1}^{n} c_i x_i^{\text{DYN}}$.*

Proof. We have

$$\begin{aligned}
\phi_i &= c_i x_i^{\text{DYN}} \cdot \ln \frac{(1+\delta)x_i^{\text{DYN}}}{x_i + \delta x_i^{\text{DYN}}} \\
&\leq c_i x_i^{\text{DYN}} \cdot \ln \frac{(1+\delta)x_i^{\text{DYN}}}{\delta x_i^{\text{DYN}}} \\
&= c_i x_i^{\text{DYN}} \cdot \ln \left(1 + \frac{1}{\delta}\right) \\
&= c_i x_i^{\text{DYN}} \cdot O(\log k).
\end{aligned}$$

□

Finally, we bound the rate of decrease of potential ϕ with increase in the variables x_i in Algorithm 1. Our goal is to show that up to constant factors, the decrease in potential ϕ can pay for the increase in cost of the solution of Algorithm 1.

Lemma 3.4. *The rate of decrease of the potential ϕ with increase in the variables x_i in Algorithm 1 is at least $\frac{1}{2}$.*

Proof. Recall that $\phi_i = c_i \cdot x_i^{\text{DYN}} \cdot \ln \frac{(1+\delta)x_i^{\text{DYN}}}{x_i + \delta x_i^{\text{DYN}}}$. Therefore,

$$\frac{d\phi_i}{dx_i} = -c_i \cdot \frac{x_i^{\text{DYN}}}{x_i + \delta x_i^{\text{DYN}}}.$$

Recall that in Algorithm 1, the rate of increase of variables x_i is given by $\frac{dx_i}{dt} = \frac{a_{ij}}{c_i}(x_i + \delta \cdot x_{ij})$, where $\delta = \frac{1}{k}$ and $x_{ij} = \sum_{s=1}^{k} x_i(j,s)$. Thus, we have:

$$\begin{aligned}
\frac{d\phi_i}{dt} = \frac{d\phi_i}{dx_i} \cdot \frac{dx_i}{dt} &= -c_i \cdot \frac{x_i^{\text{DYN}}}{x_i + \delta x_i^{\text{DYN}}} \cdot \frac{a_{ij}}{c_i}(x_i + \delta \cdot x_{ij}) \\
&= -a_{ij} \cdot x_i^{\text{DYN}} \cdot \frac{x_i + \delta x_{ij}}{x_i + \delta x_i^{\text{DYN}}}.
\end{aligned}$$

Now, we have two cases:

- If $x_{ij} \geq x_i^{\text{DYN}}$, then

$$\begin{aligned}
\frac{d\phi_i}{dt} &= -a_{ij} \cdot x_i^{\text{DYN}} \cdot \frac{x_i + \delta x_{ij}}{x_i + \delta x_i^{\text{DYN}}} \\
&\leq - a_{ij} \cdot x_i^{\text{DYN}} \cdot \frac{x_i + \delta x_i^{\text{DYN}}}{x_i + \delta x_i^{\text{DYN}}} = -a_{ij} x_i^{\text{DYN}}. \qquad (2)
\end{aligned}$$

- If $x_{ij} < x_i^{\text{DYN}}$, then

$$\begin{aligned}
\frac{d\phi_i}{dt} &= -a_{ij} \cdot x_i^{\text{DYN}} \cdot \frac{x_i + \delta \cdot x_{ij}}{x_i + \delta x_i^{\text{DYN}}} \\
&= - a_{ij} \cdot x_{ij} \cdot \frac{\frac{x_i^{\text{DYN}}}{x_{ij}} \cdot x_i + \delta \cdot x_i^{\text{DYN}}}{x_i + \delta x_i^{\text{DYN}}} < -a_{ij} x_{ij}. \qquad (3)
\end{aligned}$$

We know that at least one of the suggestions in the j-th step is supported by the DYNAMIC benchmark. Let $s(j) \in [k]$ be such a supported suggestion. Then,

$$x_{ij} = \sum_{s=1}^{k} x_i(j,s) \geq x_i(j, s(j)), \text{ and}$$
$$x_i^{\text{DYN}} \geq x_i(j, s(j)) \text{ since } s(j) \text{ is supported by DYNAMIC.}$$

Therefore, in both cases (Equation (2) and Equation (3)) above, we get

$$\frac{d\phi_i}{dt} \leq -a_{ij} x_i(j, s(j)).$$

Let us denote $I_j = \{i \mid x_i(j, s(j)) \geq x_i\}$. Then, the total decrease in potential is given by:

$$\frac{d\phi}{dt} = \sum_{i : x_i^{\text{DYN}} \geq x_i} \frac{d\phi_i}{dt} \leq -\sum_{i \in I_j} a_{ij} x_i(j, s(j)). \qquad (4)$$

By feasibility of the $s(j)$-th suggestion for the j-th constraint, we have $\sum_{i=1}^{n} a_{ij} x_i(j, s(j)) \geq 1$. Therefore,

$$\sum_{i \in I_j} a_{ij} x_i(j, s(j)) + \sum_{i \notin I_j} a_{ij} x_i(j, s(j)) \geq 1$$

$$\text{i.e., } \sum_{i \in I_j} a_{ij} x_i(j, s(j)) + \sum_{i \notin I_j} a_{ij} x_i > 1,$$

since $x_i > x_i(j, j(s))$ for $i \notin I_j$. Thus,

$$\sum_{i \in I_j} a_{ij} x_i(j, s(j)) + \sum_{i} a_{ij} x_i > 1$$

$$\text{i.e., } \sum_{i \in I_j} a_{ij} x_i(j, s(j)) > \frac{1}{2},$$

since $\sum_{i=1}^{n} a_{ij} x_i < \frac{1}{2}$ in Algorithm 1. The lemma follows by Equation (4). $\square$

Theorem 2.1 now follows from the above lemmas using standard arguments as follows:

Proof of Theorem 2.1. Initially, let $x_i = 0$ for all $i \in [n]$ but let x_i^{DYN} be their final value. Then, by Lemma 3.3, the potential ϕ is at most $O(\log k)$ times the cost of DYNAMIC. Now, as Algorithm 1 increases the values of the variables x_i, it incurs cost at rate at most $\frac{3}{2}$ (by Lemma 3.1) and the potential ϕ decreases at rate at least $\frac{1}{2}$ (by Lemma 3.4). Since ϕ is always non-negative (by Lemma 3.2), it follows that the total cost of the algorithm is at most 3 times the potential ϕ at the beginning, i.e., at most $O(\log k)$ times the DYNAMIC benchmark. This completes the proof of Theorem 2.1. $\square$

3.1. Robust Algorithm for the Online Covering Problem

Now, we prove the robust version of Theorem 2.1, namely Theorem 2.3.

Proof of Theorem 2.3. We run a meta algorithm with two sets of suggestions corresponding to two solutions. The first solution is obtained by using Algorithm 1 with k suggestions. By Theorem 2.1 this solution has cost at most $O(\log k) \cdot$ DYNAMIC. The second solution is produced by the online algorithm that achieves a competitive ratio of α in the statement of Theorem 2.3. Using Algorithm 1 again for the meta algorithm, Theorem 2.3 now follows by invoking Theorem 2.1. $\square$

As one particular application of Theorem 2.3, we note that for general OCP, the best competitive ratio is due to the following result of Gupta and Nagarajan (Gupta & Nagarajan, 2014) (see also Buchbinder and Naor (Buchbinder & Naor, 2009a)):

Theorem 3.5 (Gupta and Nagarajan (Gupta & Nagarajan, 2014))**.** *There is an algorithm for the online covering problem that has a competitive ratio of $O(\log d)$, where d is the maximum number of variables with non-zero coefficients in any constraint.*

This automatically implies the following corollary of Theorem 2.3:

Theorem 3.6. *There is an algorithm for the fractional online covering problem that produces an online solution whose cost is at most $O(\min\{\log k \cdot \text{DYNAMIC}, \ln d \cdot \text{OPT}\})$ in the multiple predictions setting with k predictions, where d is the maximum number of non-zero coefficients in any constraints of the online covering problem instance.*

For specific problems that can be modeled as OCP, it might be possible to obtain a better competitive ratio than $O(\log d)$ by using the structure of those instances. In that case, the competitive ratio in the multiple predictions setting also improves accordingly by Theorem 2.3.

3.2. Lower Bound for the Online Covering Problem

Here we show that the competitive ratio obtained in Theorem 2.1 is tight, i.e., we prove Theorem 2.2. We will restrict ourselves to instances of OCP where $a_{ij} \in \{0, 1\}$ and $c_i = 1$ for all i, j; this is called the *online (fractional) set cover* problem. (We will discuss the set cover problem in more detail in the next section.) Moreover, our lower bound will hold even in the experts model, i.e., against the STATIC benchmark. Since the DYNAMIC benchmark is always at most the STATIC benchmark, it follows that the lower bound also holds for the DYNAMIC benchmark.

Proof of Theorem 2.2. We index the k experts $\{1, 2 \ldots k\}$ using a uniform random permutation. We will construct an instance of OCP, where the cost of the optimal solution is T. The instance has k rounds, where in each round there are T constraints. We index the jth constraint of the ith round as (i, j) for $i \in [k], j \in [T]$. There are kT variables that are also indexed as (i, j) for $i \in [k], j \in [T]$. Constraint (i, j) is satisfied by each of the variables (i', j) for all $i' \geq i$ (i.e., $a_{(i,j),(i',j)} = 1$). When constraint (i, j) is presented (in round i), expert i' for every $i' \geq i$ sets variable (i', j) to 1 to satisfy it. (The suggestions of experts $i'' < i$ are immaterial in this round, and they can set any arbitrary variable satisfying constraint (i, j) to 1.)

The optimal solution is to follow expert k, i.e., the variables (k, j) for all $j \in [T]$ should be set to 1; this has cost T. After round i, the cumulative expected cost of any deterministic algorithm across the variables $(i, 1), (i, 2), \ldots, (i, T)$ is at least $\frac{T}{i}$. Across all $i \in [k]$, this adds up to a total expected cost $T \cdot \left(1 + \frac{1}{2} + \ldots + \frac{1}{k}\right) = \Omega(T \log k)$. The theorem then follows by Yao's minimax principle (Yao, 1977). $\square$

4. Online Set Cover

In the (weighted) set cover problem, we are given a collection of subsets $\mathcal{S}$ of a ground set U, where set $S \in \mathcal{S}$ has weight w_S. The goal is to select a collection of sets $\mathcal{T} \subseteq \mathcal{S}$ of minimum weight $\sum_{T\in\mathcal{T}} w_T$ that cover all the elements in U, i.e., that satisfies $\cup_{T\in\mathcal{T}} T = U$. In the *online* version of this problem (see (Alon et al., 2009)), the set of elements in U is not known in advance. In each online step, a new element $u \in U$ is revealed, and the sets in $\mathcal{S}$ that contain u are identified. If u is not covered by the sets in the current solution $\mathcal{T}$, then the algorithm must *augment* $\mathcal{T}$ by adding some set containing u to it.

In the fractional version, sets can be selected to fractions in $[0, 1]$, i.e., a solution is given by variables $x_S \in [0, 1]$ for all $S \in \mathcal{S}$. The constraint is that the total fraction of all sets containing each element u must be at least 1, i.e., $\sum_{S:u\in S} x_S \geq 1$ for every element $u \in U$. The cost of the solution is given by $\sum_{S\in\mathcal{S}} w_S x_S$. Clearly, this is a special case of the online covering problem in the previous section where each variable x_i represents a set and each constraint $\sum_{i=1}^{n} a_{ij} x_i \geq 1$ is for an element, where $a_{ij} = 1$ if and only if element j is in set i, else $a_{ij} = 0$.

The *frequency* of an element is the number of sets containing it. Let us denote the maximum frequency of any element by d. Note that this coincides with the maximum number of non-zero coefficients in any constraint. The following theorem is an immediate corollary of Theorem 3.6:

Theorem 4.1. *There is an algorithm for the fractional online set cover problem that produces an online solution whose cost is at most $O(\min\{\ln k \cdot \text{DYNAMIC}, \ln d \cdot \text{OPT}\})$ in the multiple predictions setting with k predictions.*

It is interesting to note that when the suggestions are good in the sense that $\text{DYNAMIC} = O(\text{OPT})$, the competitive ratio of $O(\log k)$ in the above theorem is independent of the size of the problem instance. In contrast, for the classical fractional online set cover problem, there is a well-known lower bound of $\Omega(\log d)$ on the competitive ratio. We also note that the competitive ratio in Theorem 4.1 is tight since as we noted earlier, the lower bound instance constructed in Theorem 2.2 is actually an instance of the set cover problem.

Theorem 4.1 obtains a fractional solution to the set cover problem that can be rounded online to obtain an integer solution using standard techniques. Details of the integral case are given in Appendix A.

5. (Weighted) Caching

The caching problem is among the most well-studied online problems (see, e.g., (Sleator & Tarjan, 1985; Fiat et al., 1991; McGeoch & Sleator, 1991; Achlioptas et al., 2000; Young, 1991; Blum et al., 1999), and the textbook (Borodin & El-Yaniv, 1998)). In this problem, there is a set of n pages and a cache that can hold any h pages at a time.[3] In every online step j, a page p_j is requested; if this page is not in the cache, then it has to be brought into the cache (called *fetching*) by evicting an existing page from the cache. In the weighted version (see, e.g., (Chrobak et al., 1991; Young, 1994; Bansal et al., 2007)), the cost of fetching a page p into the cache is given by its non-negative weight w_p. (In the unweighted version, $w_p = 1$ for all pages.) The goal of a caching algorithm is to minimize the total cost of fetching pages while serving all requests.

The (weighted) caching problem can be formulated as online covering problem by defining variables $x_p(r) \in \{0, 1\}$ to indicate whether page p is evicted between its r-th and $(r + 1)$-st requests. Let $r(p, j)$ denote the number of times page p is requested until (and including) the j-th request. For any online step j, let $B(j) = \{p : r(p, j) \geq 1\}$ denote the set of pages that have been requested until (and including) the j-th request. The covering program formulation is as follows:

$$\begin{array}{lll} \min & \sum_p \sum_r w_p \cdot x_p(r) \text{ subject to} & \\ & \sum_{p\in B(j), p\neq p_j} x_p(r(p, j)) \geq |B(j)| - h, \forall j \geq 1 & \\ & x_p(r) \in \{0, 1\}, & \forall p, \forall r \geq 1 \end{array}$$

In the fractional version of the problem, we replace the constraints $x_p(r) \in \{0, 1\}$ with the constraints $x_p(r) \in [0, 1]$. Clearly, this fits the definition of the online covering problem.[4] Moreover, for the fractional weighted caching problem, Bansal, Buchbinder, and Naor gave an online algorithm with a competitive ratio of $O(\log h)$:

Theorem 5.1 ((Bansal et al., 2007))**.** *There is an online algorithm for the fractional weighted caching problem with a competitive ratio of $O(\log h)$.*

Note that the competitive ratio of $O(\log h)$ is better than that given by Theorem 3.5 since the cache size h is typically much smaller than the total number of pages. Now, we apply Theorem 2.3 to get the following result for fractional weighted caching with k predictions:

Theorem 5.2. *There is an algorithm for the fractional weighted caching problem that produces an online solution whose cost is at most $O(\min\{\ln k \cdot \text{DYNAMIC}, \ln h \cdot \text{OPT}\})$ in the multiple predictions setting with k predictions.*

As for set cover, the fractional solution for weighted caching can also be rounded online to obtain an integer solution (see Appendix A).

[3]The usual notation for cache size is k, but we have changed it to h since we are using k to denote the number of suggestions.

[4]Strictly speaking, we need to scale the first set of coefficients by $|B(j)| - h$, but as we mentioned earlier, this is equivalent since scaling the coefficients has no bearing on the competitive ratio of our algorithm.

6. Online Covering with Box Constraints

In this section, we generalize the online covering framework in Section 2 by allowing additional *box*-type constraints of the form $x_{ij} \leq y_i$. The new linear program is given by:

$$\text{Minimize} \sum_{i=1}^{n} c_i y_i + \sum_{i=1}^{n}\sum_{j=1}^{m} d_{ij} x_{ij} \text{ subject to}$$

$$x_{ij} \leq y_i \quad \forall i, j \tag{5}$$

$$\sum_{i=1}^{n} a_{ij} x_{ij} \geq 1 \quad \forall j \tag{6}$$

$$y_i \in [0,1] \quad \forall i \tag{7}$$

The box constraints $x_{ij} \leq y_i$ do not have coefficients and hence are known offline. As in OCP, the cost coefficients c_i are known offline. In each online step, a new covering constraint $\sum_{i=1}^{n} a_{ij} x_{ij} \geq 1$ is revealed to the algorithm, and the corresponding costs of coefficients d_{ij} are also revealed.

We first note that this is a generalization of the online covering problem. To see this, set $d_{ij} = 0$. Then, we get:

$$\min_{y_i \in [0,1]: i \in [n]} \left\{ \sum_{i=1}^{n} c_i y_i : \sum_{i=1}^{n} a_{ij} y_i \geq 1 \, \forall j \in [m] \right\},$$

which is precisely the online covering problem.

The more general version captures problems like facility location (shown in the next section) that are not directly modeled by OCP. We denote the s-th suggestion for the j-th constraint by variables $y_i(j, s)$ and $x_{ij}(s)$; they satisfy $\sum_{i=1}^{n} a_{ij} x_{ij}(s) \geq 1$, i.e., all suggestions are feasible.

For this more general problem, we obtain the following theorem that is an analog of Theorem 2.3.

Theorem 6.1. *Suppose a class of online covering problems with box constraints have an online algorithm (without predictions) whose competitive ratio is α. Then, there is an algorithm for this class of online problems with k suggestions that produces an online solution whose cost is at most $O(\min\{\log k \cdot \text{DYNAMIC}, \alpha \cdot \text{OPT}\})$.*

The proof of this theorem (Appendix B) is similar to OCP.

7. Online Facility Location

We apply the online covering with box constraints framework to the online FACILITY LOCATION problem. An instance of FACILITY LOCATION is given by a set of potential facility locations $\mathcal{F}$, a set $\mathcal{R}$ of client locations and a metric space $d(\cdot, \cdot)$ defined on these points. Further, each location $f_i \in \mathcal{F}$ has a facility opening cost o_i. A solution opens a subset $\mathcal{F}' \subseteq \mathcal{F}$ of facilities, and assigns each client r_j to the closest open facility $f_{i_j} \in \mathcal{F}'$. The connection cost of this client is $d(r_j, f_{i_j})$ and the goal is to minimize the sum of the connection costs of all the clients and the facility opening costs of the facilities in $\mathcal{F}'$.

In the online FACILITY LOCATION problem, clients arrive over time and the solution needs to assign each arriving client to an open facility (possibly after opening a new facility). This problem was first studied by (Meyerson, 2001) who gave a competitive ratio of $O(\log n)$ for n clients. This was later improved by (Fotakis, 2008) to $O(\frac{\log n}{\log \log n})$. Since then, many variants of the problem have been studied (see survey (Fotakis, 2011)). We will encode this problem in the online covering (with box constraints) framework. We have a variable y_i for each facility location $f_i \in \mathcal{F}$, which denotes the extent to which facility f_i is open, and a variable x_{ij} for each client r_j and facility f_i denoting the extent to which r_j is assigned to f_i. Note that this is a special case of the online covering problem with box constraints, where $c_i = o_i$ for all i, and $a_{ij} = 1$ for all i, j. As a corollary of Theorem 6.1, we get the following result:

Theorem 7.1. *There is an algorithm for the fractional online FACILITY LOCATION problem that produces an online solution with cost $O(\min\{\frac{\log n}{\log \log n} \cdot \text{OPT}, \log k \cdot \text{DYNAMIC}\})$ in the multiple predictions setting with k predictions.*

We also note that the $O(\log k)$ bound in Theorem 7.1 cannot be improved further (except lower order terms) since we can simulate an instance of the online facility location problem without predictions by giving all prior client locations as suggested facility locations for $k = n$. Furthermore, a lower bound of $\Omega\left(\frac{\log n}{\log \log n}\right)$ is known (due to Fotakis (Fotakis, 2008)) for online facility location, and this construction easily extends to the fractional version of the problem.

8. Conclusion

This paper presented a general recipe for the design of online algorithms with multiple ML predictions for covering problems, and applied it to set cover, (weighted) caching, and facility location. It would be interesting to extend this framework to packing problems such as online budgeted allocation, and to more general settings for mixed packing and covering LPs and non-linear (convex) objectives.

9. Acknowledgements

This research was supported by the Indo-US Virtual Networked Joint Center No. IUSSTF/JC-017/2017. K. Anand and D. Panigrahi were supported in part by NSF Awards CCF-1955703 and CCF-1750140 (CAREER), and ARO Award W911NF2110230. R. Ge was supported in part by NSF Awards DMS-2031849, CCF-1704656, CCF-1845171 (CAREER), CCF-1934964 (TRIPODS), a Sloan Research Fellowship, and a Google Faculty Research Award.

References

Achlioptas, D., Chrobak, M., and Noga, J. Competitive analysis of randomized paging algorithms. *Theor. Comput. Sci.*, 234(1-2):203–218, 2000.

Almanza, M., Chierichetti, F., Lattanzi, S., Panconesi, A., and Re, G. Online facility location with multiple advice. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*, 2021.

Alon, N., Awerbuch, B., Azar, Y., Buchbinder, N., and Naor, J. A general approach to online network optimization problems. *ACM Transactions on Algorithms*, 2(4):640–660, 2006.

Alon, N., Awerbuch, B., Azar, Y., Buchbinder, N., and Naor, J. The online set cover problem. *SIAM J. Comput.*, 39(2): 361–370, 2009.

Anand, K., Ge, R., and Panigrahi, D. Customizing ML predictions for online algorithms. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 303–313. PMLR, 2020.

Anand, K., Ge, R., Kumar, A., and Panigrahi, D. A regression approach to learning-augmented online algorithms. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021.*, 2021.

Antoniadis, A., Coester, C., Elias, M., Polak, A., and Simon, B. Online metric algorithms with untrusted predictions. In *Proceedings of the 37th International Conference on Machine Learning,ICML 2020*, 2020a.

Antoniadis, A., Gouleakis, T., Kleer, P., and Kolev, P. Secretary and online matching problems with machine learned advice. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020b.

Azar, Y., Bhaskar, U., Fleischer, L., and Panigrahi, D. Online mixed packing and covering. In Khanna, S. (ed.), *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pp. 85–100. SIAM, 2013.

Azar, Y., Buchbinder, N., Chan, T. H., Chen, S., Cohen, I. R., Gupta, A., Huang, Z., Kang, N., Nagarajan, V., Naor, J., and Panigrahi, D. Online algorithms for covering and packing problems with convex objectives. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pp. 148–157, 2016.

Azar, Y., Leonardi, S., and Touitou, N. Flow time scheduling with uncertain processing time. In Khuller, S. and Williams, V. V. (eds.), *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pp. 1070–1080. ACM, 2021.

Azar, Y., Panigrahi, D., and Touitou, N. Online graph algorithms with predictions. In *Proceedings of the Thirty-third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, January 9-12*, 2022.

Bamas, É., Maggiori, A., Rohwedder, L., and Svensson, O. Learning augmented energy minimization via speed scaling. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020a.

Bamas, É., Maggiori, A., and Svensson, O. The primal-dual method for learning augmented algorithms. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020b.

Bansal, N. and Coester, C. Online metric allocation. *CoRR*, abs/2111.15169, 2021. URL `https://arxiv.org/abs/2111.15169`.

Bansal, N., Buchbinder, N., and Naor, J. A primal-dual randomized algorithm for weighted paging. In *FOCS*, pp. 507–517, 2007.

Bansal, N., Buchbinder, N., and Naor, J. S. A simple analysis for randomized online weighted paging. *Unpublished Manuscript*, 2010. URL `https://www.win.tue.nl/~nikhil/pubs/pot-wt2.pdf`.

Bansal, N., Coester, C., Kumar, R., Purohit, M., and Vee, E. Scale-free allocation, amortized convexity, and myopic weighted paging. *CoRR*, abs/2011.09076, 2020. URL `https://arxiv.org/abs/2011.09076`.

Bhaskara, A., Cutkosky, A., Kumar, R., and Purohit, M. Online learning with imperfect hints. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 822–831. PMLR, 2020.

Blum, A., Burch, C., and Kalai, A. Finely-competitive paging. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pp. 450–458. IEEE Computer Society, 1999.

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

Buchbinder, N. and Naor, J. Online primal-dual algorithms for covering and packing. *Math. Oper. Res.*, 34(2):270–286, 2009a.

Buchbinder, N. and Naor, J. The design of competitive online algorithms via a primal-dual approach. *Foundations and Trends in Theoretical Computer Science*, 3(2-3): 93–263, 2009b.

Buchbinder, N., Jain, K., and Naor, J. Online primal-dual algorithms for maximizing ad-auctions revenue. In *ESA*, pp. 253–264, 2007.

Buchbinder, N., Gupta, A., Molinaro, M., and Naor, J. S. k-servers with a smile: Online algorithms via projections. In Chan, T. M. (ed.), *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pp. 98–116. SIAM, 2019.

Chrobak, M., Karloof, H., Payne, T., and Vishwnathan, S. New results on server problems. *SIAM Journal on Discrete Mathematics*, 4(2):172–181, 1991.

Dütting, P., Lattanzi, S., Leme, R. P., and Vassilvitskii, S. Secretaries with advice. In Biró, P., Chawla, S., and Echenique, F. (eds.), *EC '21: The 22nd ACM Conference on Economics and Computation, Budapest, Hungary, July 18-23, 2021*, pp. 409–429. ACM, 2021.

Eden, T., Indyk, P., Narayanan, S., Rubinfeld, R., Silwal, S., and Wagner, T. Learning-based support estimation in sublinear time. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Fiat, A., Karp, R. M., Luby, M., McGeoch, L. A., Sleator, D. D., and Young, N. E. Competitive paging algorithms. *J. Algorithms*, 12(4):685–699, 1991.

Fotakis, D. On the competitive ratio for online facility location. *Algorithmica*, 50(1):1–57, 2008.

Fotakis, D. Online and incremental algorithms for facility location. *SIGACT News*, 42(1):97–131, 2011.

Fotakis, D., Gergatsouli, E., Gouleakis, T., and Patris, N. Learning augmented online facility location. *CoRR*, abs/2107.08277, 2021. URL https://arxiv.org/abs/2107.08277.

Gollapudi, S. and Panigrahi, D. Online algorithms for rent-or-buy with expert advice. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2319–2327. PMLR, 2019.

Gupta, A. and Nagarajan, V. Approximating sparse covering integer programs online. *Math. Oper. Res.*, 39(4):998–1011, 2014.

Hsu, C., Indyk, P., Katabi, D., and Vakilian, A. Learning-based frequency estimation algorithms. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Indyk, P., Vakilian, A., and Yuan, Y. Learning-based low-rank approximations. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 7400–7410, 2019.

Jiang, S. H., Liu, E., Lyu, Y., Tang, Z. G., and Zhang, Y. Online facility location with predictions. *CoRR*, abs/2110.08840, 2021a. URL https://arxiv.org/abs/2110.08840.

Jiang, Z., Panigrahi, D., and Sun, K. Online algorithms for weighted caching with predictions. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020*, 2020.

Jiang, Z., Lu, P., Tang, Z. G., and Zhang, Y. Online selection problems against constrained adversary. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5002–5012. PMLR, 2021b.

Khanafer, A., Kodialam, M., and Puttaswamy, K. P. N. The constrained ski-rental problem and its application to online cloud cost optimization. In *Proceedings of the INFOCOM*, pp. 1492–1500, 2013.

Kumar, R., Purohit, M., and Svitkina, Z. Improving online algorithms via ML predictions. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 9684–9693, 2018.

Lattanzi, S., Lavastida, T., Moseley, B., and Vassilvitskii, S. Online scheduling via learned weights. In Chawla, S. (ed.), *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pp. 1859–1877. SIAM, 2020.

Lavastida, T., Moseley, B., Ravi, R., and Xu, C. Learnable and instance-robust predictions for online matching, flows and load balancing. In Mutzel, P., Pagh, R., and Herman, G. (eds.), *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference)*, volume 204 of *LIPIcs*, pp. 59:1–59:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

Lee, R., Maghakian, J., Hajiesmaili, M. H., Li, J., Sitaraman, R. K., and Liu, Z. Online peak-aware energy scheduling with untrusted advice. In de Meer, H. and Meo, M. (eds.), *e-Energy '21: The Twelfth ACM International Conference on Future Energy Systems, Virtual Event, Torino, Italy, 28 June - 2 July, 2021*, pp. 107–123. ACM, 2021.

Lykouris, T. and Vassilvitskii, S. Competitive caching with machine learned advice. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3302–3311. PMLR, 2018.

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

McGeoch, L. A. and Sleator, D. D. A strongly competitive randomized paging algorithm. *Algorithmica*, 6(6):816–825, 1991.

Meyerson, A. Online facility location. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pp. 426–431. IEEE Computer Society, 2001.

Mitzenmacher, M. A model for learned bloom filters and optimizing by sandwiching. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 462–471, 2018.

Mitzenmacher, M. Scheduling with predictions and the price of misprediction. In Vidick, T. (ed.), *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pp. 14:1–14:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

Shalev-Shwartz, S. Online learning and online convex optimization. *Found. Trends Mach. Learn.*, 4(2):107–194, 2012.

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

Wang, S., Li, J., and Wang, S. Online algorithms for multi-shop ski rental with machine learned advice. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Wei, A. Better and simpler learning-augmented online caching. In Byrka, J. and Meka, R. (eds.), *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*, volume 176 of *LIPIcs*, pp. 60:1–60:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

Wei, A. and Zhang, F. Optimal robustness-consistency trade-offs for learning-augmented online algorithms. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Yao, A. C.-C. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pp. 222–227. IEEE Computer Society, 1977.

Young, N. On-line caching as cache size varies. In *Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '91, pp. 241–250, Philadelphia, PA, USA, 1991. Society for Industrial and Applied Mathematics. ISBN 0-89791-376-0.

Young, N. E. The k-server dual and loose competitiveness for paging. *Algorithmica*, 11(6):525–541, 1994.

A. Integral Solutions via Online Rounding

In this section, we combine the fractional solutions produced by the algorithms described earlier with existing online rounding techniques to obtain integral solutions for problems. In general, we state the following meta theorem:

Theorem A.1. *Suppose there is an online rounding algorithm for a class of online covering problems (with box constraints) such that the cost of (output) integral solution is at most β times the cost of the (input) fractional solution. Moreover, suppose there is an online algorithm (without predictions) for this class of online covering problems that produces integral solutions and has a competitive ratio of α. Then, there is an algorithm for this class of online covering problems with k suggestions that produces an online integral solution whose cost is at most $O(\min\{\beta \ln k \cdot \textsc{dynamic}, \alpha \cdot \textsc{opt}\})$.*

We now consider some specific instantiations of this meta theorem for specific integral covering problems.

A.1. Integral Set Cover

We consider the integral set cover problem, i.e., where the variables x_i are required to be integral, i.e., in $\{0, 1\}$ rather than in $[0, 1]$. The following a well-known result on rounding fractional set cover solutions online:

Theorem A.2 (Alon *et al.* (Alon et al., 2009))**.** *Given any feasible solution to the fractional online set cover problem, there is an online algorithm for finding a feasible solution to the integral online set cover problem whose cost is at most $O(\log m)$ times that of the fractional solution, where m is the number of elements.*

By applying this theorem on the fractional solution produced by Theorem 4.1, we get a competitive ratio of $O(\log m \log k)$ for the integral online set cover problem with k predictions against the $\textsc{dynamic}$ benchmark:

Theorem A.3. *There is an algorithm for the integral online set cover problem that produces an online solution whose cost is at most $O(\min\{\ln m \ln k \cdot \textsc{dynamic}, \ln m \ln d \cdot \textsc{opt}\})$ in the multiple predictions setting with k predictions.*

It is well-known that the competitive ratio of the integral online set cover problem (without predictions) is at least $\tilde{\Omega}(\log m \log d)$ (Alon et al., 2009)[5]. In the degenerate case of $k = d$, any instance of online set cover can be generated in the k predictions settings where the benchmark solution $\textsc{dynamic}$ will be the optimal solution, since all the sets containing an element can be provided as suggestions in each online step. As a consequence, the competitive ratio in Theorem A.3 is tight (up to lower order terms).

A.2. Integral (Weighted) Caching

Next, we consider the integral weighted caching problem, i.e., where the variables $x_p(r)$ have to be in $\{0, 1\}$ rather than $[0, 1]$. We use the following known result about online rounding of fractional weighted caching solutions:

Theorem A.4 (Bansal, Buchbinder, and Naor (Bansal et al., 2007))**.** *Given any feasible online solution to the fractional weighted caching problem, there is an online algorithm for finding a feasible online solution to the integral weighted caching problem whose cost is at most $O(1)$ times that of the fractional solution.*

By applying this theorem on the fractional solution produced by Theorem 5.2, we get a competitive ratio of $O(\log k)$ for the integral weighted caching problem with k predictions against the $\textsc{dynamic}$ benchmark:

Theorem A.5. *There is an online algorithm for the integral weighted caching problem that produces an online solution whose cost is at most $O(\min\{\ln k \cdot \textsc{dynamic}, \ln h \cdot \textsc{opt}\})$ in the multiple predictions setting with k predictions.*

B. Details for Online Covering with Box Constraints

In this section, we give details of the online covering problem with box constraints, and prove Theorem 6.1.

B.1. Online Algorithm

Our algorithm for $\textsc{ocp}$ with box constraints is given in Algorithm 2. As earlier, for all i, we simultaneously raise x_{ij} (and possibly y_i as well) at the rate specified by the algorithm. As in the case of $\textsc{ocp}$, we raise these variables only till the constraint is satisfied up to a factor of $\frac{1}{2}$, and any individual variable does not cross $\frac{1}{2}$. This allows us to double all variable

[5] The notations $\tilde{\Omega}(\cdot)$ and $\tilde{O}(\cdot)$ hide lower order terms.

and satisfy the constraints at an additional factor of 2 in the cost. Moreover, the same argument as in Algorithm 1 implies that this algorithm is also feasible, i.e., there is at least one variable that can be increased in the **while** loop.

Algorithm 2 Algorithm for Online FACILITY LOCATION

On arrival of a new constraint $\sum_{i=1}^{n} a_{ij} x_{ij} = 1$:
  Initialize $x_{ij} = 0, \quad \forall j$.
  Set $\Gamma_{ij} := \sum_{s=1}^{k} x_{ij}(s), \quad \forall j$ and $\delta = \frac{1}{k}$.
    while $\sum_j x_{ij} < \frac{1}{2}$
      for all i such that $x_{ij} < \frac{1}{2}$:
        if $x_{ij} < y_i$
          Increase x_{ij} at the rate $\frac{\partial x_{ij}}{\partial t} = \left(\frac{a_{ij}}{d_{ij}}\right) \cdot (x_{ij} + \delta \cdot \Gamma_{ij})$
        else
          Increase both variables x_{ij}, y_i at the same rate
          $\frac{\partial y_i}{\partial t} = \frac{\partial x_{ij}}{\partial t} = \left(\frac{a_{ij}}{d_{ij}+c_i}\right) \cdot (x_{ij} + \delta \cdot \Gamma_{ij})$.

B.2. Analysis

In this section, we analyze Algorithm 2. We first bound the rate of increase of the cost of the algorithm.

Lemma B.1. *The rate of increase of the cost for Algorithm 2 is at most $\frac{3}{2}$.*

Proof. Consider Algorithm 2 when the constraint $\sum_{i=1}^{n} a_{ij} x_{ij} = 1$ arrives. For any i, we claim:

$$d_{ij} \cdot \frac{\partial x_{ij}}{\partial t} + c_i \cdot \frac{\partial y_i}{\partial t} = a_{ij} \left(x_{ij} + \delta \cdot \Gamma_{ij}\right). \tag{8}$$

To show this, there are two cases to consider:

- $x_{ij} < y_i$: In this case, $\frac{\partial y_i}{\partial t} = 0$, and $d_{ij} \cdot \frac{\partial x_{ij}}{\partial t} = a_{ij} \left(x_{ij} + \delta \cdot \Gamma_{ij}\right)$ and so (8) holds.
- $x_{ij} = y_i$: In this case, $\frac{\partial y_i}{\partial t} = \frac{\partial x_{ij}}{\partial t} = \frac{a_{ij}}{d_{ij}+c_i} \cdot \left(x_{ij} + \frac{\Gamma_{ij}}{k}\right)$, i.e., $c_i \cdot \frac{\partial y_i}{\partial t} + d_{ij} \cdot \frac{\partial x_{ij}}{\partial t} = a_{ij} \left(x_{ij} + \delta \cdot \Gamma_{ij}\right)$, and so (8) holds.

Therefore, the rate of change of the objective function is given by:

$$\sum_i \left(d_{ij} \cdot \frac{\partial x_{ij}}{\partial t} + c_i \cdot \frac{\partial y_i}{\partial t}\right) \stackrel{(8)}{=} \sum_i a_{ij} \left(x_{ij} + \delta \cdot \Gamma_{ij}\right) = \sum_i a_{ij} x_{ij} + \frac{\sum_s \sum_i a_{ij} x_{ij}(s)}{k} \leq \frac{1}{2} + 1 = \frac{3}{2}. \qquad \square$$

We now describe the potential function, which has a similar structure as OCP. Let $x_{ij}^{\text{DYN}}, y_i^{\text{DYN}}$ denote the values of the variables x_{ij}, y_i respectively in the benchmark solution DYNAMIC. The potential function for a variable x_{ij} is then defined as follows:

$$\phi_{ij} = d_{ij} \cdot x_{ij}^{\text{DYN}} \cdot \ln \frac{(1+\delta) x_{ij}^{\text{DYN}}}{x_{ij} + \delta x_{ij}^{\text{DYN}}}, \text{ where } \delta = \frac{1}{k},$$

and the potential for the variable y_i is given by:

$$\psi_i = c_i \cdot y_i^{\text{DYN}} \cdot \ln \frac{(1+\delta) y_i^{\text{DYN}}}{y_i + \delta y_i^{\text{DYN}}}.$$

The overall potential is

$$\phi = \sum_{i,j: x_{ij}^{\text{DYN}} \geq x_{ij}} \phi_{ij} + \sum_{i: y_i^{\text{DYN}} \geq y_i} \psi_i.$$

The rest of the proof proceeds along the same lines as that for the online covering problem. But we give the details for the sake of completeness.

The next lemma is the analogue of Lemma 3.2, and shows that the potential ϕ is always non-negative.

Lemma B.2. *For any values of the variables $x_{ij}, y_i, x_{ij}^{\text{DYN}}, y_i^{\text{DYN}}$, the potential function ϕ is non-negative.*

Proof. We show that each of the quantities ϕ_{ij}, ψ_i in the expression for ϕ is non-negative. Consider a pair i, j for which $x_{ij}^{\text{DYN}} \geq x_{ij}$. Then

$$\phi_{ij} = d_{ij} \cdot x_{ij}^{\text{DYN}} \cdot \ln \frac{(1+\delta)x_{ij}^{\text{DYN}}}{x_{ij} + \delta x_{ij}^{\text{DYN}}} = d_{ij} \cdot x_{ij}^{\text{DYN}} \cdot \ln \frac{1+\delta}{\frac{x_{ij}}{x_{ij}^{\text{DYN}}} + \delta} \geq 0.$$

Similarly, $\psi_i \geq 0$ if $y_i^{\text{DYN}} \geq y_i$. This shows that $\phi \geq 0$. □

Now we bound the potential against the benchmark solution DYNAMIC. The proof of this lemma is very similar to that of Lemma 3.3.

Lemma B.3. *The following bounds hold: $\phi_{ij} \leq d_{ij} \cdot x_{ij}^{\text{DYN}} \cdot \ln(1+\frac{1}{\delta}) = d_{ij}x_{ij}^{\text{DYN}} \cdot O(\log k)$ and $\psi_i \leq c_i \cdot y_i^{\text{DYN}} \cdot \ln(1+\frac{1}{\delta})) = c_i y_i^{\text{DYN}} \cdot O(\log k)$. As a consequence, the overall potential $\phi \leq O(\log k) \cdot \left(\sum_i \sum_j d_{ij} x_{ij}^{\text{DYN}} + \sum_j c_i y_i^{\text{DYN}}\right)$.*

Proof. We have

$$\phi_{ij} = d_{ij}x_{ij}^{\text{DYN}} \cdot \ln \frac{(1+\delta)x_{ij}^{\text{DYN}}}{x_i + \delta x_{ij}^{\text{DYN}}} \leq d_{ij}x_{ij}^{\text{DYN}} \cdot \ln \frac{(1+\delta)x_{ij}^{\text{DYN}}}{\delta x_{ij}^{\text{DYN}}} = d_{ij}x_{ij}^{\text{DYN}} \cdot \ln\left(1 + \frac{1}{\delta}\right) = d_{ij}x_{ij}^{\text{DYN}} \cdot O(\log k).$$

The bound for ψ_i also follows similarly. □

Finally, we bound the rate of decrease of potential ϕ with increase in the variables in Algorithm 2.

Lemma B.4. *The rate of decrease of the potential ϕ in Algorithm 2 is at least $\frac{1}{2}$.*

Proof. It is easy to check that

$$\frac{\partial \phi_{ij}}{\partial x_{ij}} = -\frac{d_{ij}x_{ij}^{\text{DYN}}}{x_{ij} + \delta x_{ij}^{\text{DYN}}}, \quad \frac{\partial \psi_i}{\partial y_i} = -\frac{c_i y_i^{\text{DYN}}}{y_i + \delta y_i^{\text{DYN}}}. \tag{9}$$

Consider the step when the j-th constraint arrives. We claim that for any index $i \in [n]$,

$$\frac{\partial\left(\phi_{ij} + \psi_i\right)}{\partial t} \leq -a_{ij}x_{ij}^{\text{DYN}} \cdot \frac{x_{ij} + \delta\Gamma_{ij}}{x_{ij} + \delta x_{ij}^{\text{DYN}}}. \tag{10}$$

To prove this, we consider two cases:

- $x_{ij} < y_i$: In this case, $\frac{\partial x_{ij}}{\partial t} = \frac{a_{ij}}{d_{ij}} \cdot \left(x_{ij} + \delta\Gamma_{ij}\right), \frac{\partial y_i}{\partial t} = 0$. Combining this with (9), we see that

$$\frac{\partial\left(\phi_{ij} + \psi_i\right)}{\partial t} = \frac{\partial \phi_{ij}}{\partial t} = \frac{\partial \phi_{ij}}{\partial x_{ij}} \cdot \frac{\partial x_{ij}}{\partial t} = -a_{ij}x_{ij}^{\text{DYN}} \cdot \frac{x_{ij} + \delta\Gamma_{ij}}{x_{ij} + \delta x_{ij}^{\text{DYN}}}.$$

- $x_{ij} = y_i$: In this case, $\frac{\partial x_{ij}}{\partial t} = \frac{\partial y_i}{\partial t} = \frac{a_{ij}}{d_{ij}+c_i} \cdot \left(x_{ij} + \frac{\Gamma_{ij}}{k}\right)$. Using (9) again and the fact that $y_i = x_{ij}$, we see that

$$\frac{\partial\left(\phi_{ij} + \psi_i\right)}{\partial t} = \frac{\partial \phi_{ij}}{\partial x_{ij}} \cdot \frac{\partial x_{ij}}{\partial t} + \frac{\partial \psi_i}{\partial y_i} \cdot \frac{\partial y_i}{\partial t} = -\frac{a_{ij}}{c_i + d_{ij}} \left(\frac{d_{ij}x_{ij}^{\text{DYN}}}{x_{ij} + \delta x_{ij}^{\text{DYN}}} + \frac{c_i y_i^{\text{DYN}}}{x_{ij} + \delta y_i^{\text{DYN}}}\right) \cdot \left(x_{ij} + \delta\Gamma_{ij}\right).$$

  Since $y_i^{\text{DYN}} \geq x_{ij}^{\text{DYN}}$, $\frac{y_i^{\text{DYN}}}{x_{ij}+\delta y_i^{\text{DYN}}} \geq \frac{x_{ij}^{\text{DYN}}}{x_{ij}+\delta x_{ij}^{\text{DYN}}}$. Therefore, the RHS above is at most $-a_{ij}x_{ij}^{\text{DYN}} \cdot \frac{x_{ij}+\delta\Gamma_{ij}}{x_{ij}+\delta x_{ij}^{\text{DYN}}}$.

Thus, we have shown that inequality (10) always holds. Now, we have two cases:

- $\Gamma_{ij} \geq x_{ij}^{\text{DYN}}$: Inequality (10) implies that

$$\frac{\partial(\phi_{ij} + \psi_i)}{\partial t} \leq -a_{ij}x_{ij}^{\text{DYN}}.$$

- $\Gamma_{ij} < x_{ij}^{\text{DYN}}$: Using (10) again, we see that

$$\frac{d(\phi_{ij}+\psi_i)}{dt} \leq -a_{ij}x_{ij}^{\text{DYN}} \cdot \frac{x_{ij}+\delta\cdot\Gamma_{ij}}{x_{ij}+\delta x_{ij}^{\text{DYN}}} = -a_{ij}\Gamma_{ij}\cdot\frac{\frac{x_{ij}^{\text{DYN}}}{\Gamma_{ij}}\cdot x_{ij}+\delta\cdot x_{ij}^{\text{DYN}}}{x_{ij}+\delta x_{ij}^{\text{DYN}}} \leq -a_{ij}\Gamma_{ij}.$$

We know that at least one of the suggestions in the i-th step is supported by the DYNAMIC benchmark. Let $s(j) \in [k]$ be the index of such a supported suggestion. Then,

$$\Gamma_{ij} = \sum_{s=1}^{k} x_{ij}(s) \geq x_{ij}(s(j)), \text{ and}$$
$$x_{ij}^{\text{DYN}} \geq x_{ij}(s(j)) \text{ since } s(j) \text{ is supported by DYNAMIC.}$$

Therefore, in both cases above, we get

$$\frac{\partial(\phi_{ij}+\psi_i)}{\partial t} \leq -a_{ij}\cdot x_{ij}(s(j)).$$

Let I denote the index set $\{i\in[n] : x_{ij}^{\text{DYN}} \geq x_{ij}\}$ (here j is fixed). We claim that the total decrease in potential satisfies:

$$\frac{\partial\phi}{\partial t} \leq -\sum_{i\in I} a_{ij}\cdot x_{ij}(s(j)). \tag{11}$$

To see this, let I' denote the index set $\{i : y_i^{\text{DYN}} \geq y_i\}$. Then the term in ϕ corresponding to step j is $\phi_j := \sum_{i\in I}\phi_{ij} + \sum_{i\in I'}\psi_i$. First consider an index $i \in I$ for which $x_{ij} < y_i$. In this case, we know that $\frac{\partial\psi_i}{\partial t} = 0$, and so irrespective of whether i belongs to I' or not, the rate of change of the terms in ϕ_j corresponding to i is

$$\frac{\partial(\phi_{ij}+\psi_i)}{\partial t} \leq -a_{ij}\cdot x_{ij}(s(j)).$$

Now consider an index $i \in I$ for which $x_{ij} = y_i$. Since $y_i^{\text{DYN}} \geq x_{ij}^{\text{DYN}}$, it follows that $y_i^{\text{DYN}} \geq y_i$ and so $i \in I'$ as well. Therefore, the rate of change of the terms in ϕ_j corresponding to i is equal to

$$\frac{\partial(\phi_{ij}+\psi_i)}{\partial t} \leq -a_{ij}\cdot x_{ij}(s(j)).$$

Finally, consider an index $i \in I' \setminus I$. It is easy to verify that $\frac{\partial\psi_i}{\partial t} \leq 0$. Thus, inequality (11) follows.

The rest of the argument proceeds as in the proof of Lemma 3.4. By feasibility of the $s(j)$-th suggestion in step j, we have:

$$\begin{aligned}
\sum_{j} a_{ij}x_{ij}(s(j)) &\geq 1 \\
\text{i.e., } \sum_{i\in I} a_{ij}x_{ij}(s(j)) + \sum_{i\notin I} a_{ij}x_{ij}(s(j)) &\geq 1 \\
\text{i.e., } \sum_{i\in I} a_{ij}x_{ij}(s(j)) + \sum_{i\notin I} a_{ij}x_{ij} &> 1 \qquad \left(\text{since } i\notin I, \text{ we have } x_{ij} > x_{ij}^{\text{DYN}} \geq x_{ij}(s(j))\right) \\
\text{i.e., } \sum_{i\in I} a_{ij}x_{ij}(s(j)) + \sum_{i=1}^{n} a_{ij}x_{ij} &> 1 \\
\text{i.e., } \sum_{i\in I} a_{ij}x_{ij}(s(j)) &> \frac{1}{2} \quad \left(\text{since } \sum_{i=1}^{n} x_{ij} < \frac{1}{2} \text{ in Algorithm 2}\right).
\end{aligned}$$

The desired result now follows from (11). □

We now combine these lemmas to obtain the following result:

Theorem B.5. *There is an algorithm for the online covering problem with box constraints that produces an online solution whose cost is at most $O(\log k) \cdot$ DYNAMIC in the multiple predictions setting with k predictions.*

Proof. Initially, let $x_{ij} = y_i = 0$ for all i, j but let $x_{ij}^{\text{DYN}}, y_i^{\text{DYN}}$ be their final value. Then, by Lemma B.3, the potential ϕ is at most $O(\log k)$ times the cost of DYNAMIC. Now, as Algorithm 2 increases the values of the variables x_{ij}, y_i, it incurs cost at rate at most $\frac{3}{2}$ (by Lemma B.1) and the potential ϕ decreases at rate at least $\frac{1}{2}$ (by Lemma B.4). Since ϕ is always non-negative (by Lemma B.2), it follows that the total cost of the algorithm is at most 3 times the potential ϕ at the beginning, i.e., at most $O(\log k)$ times the DYNAMIC benchmark. □

Finally, we robustify the solution produced by Algorithm 2 using the same ideas as in Theorem 2.3. Namely, we run Algorithm 2 to produce one solution and an online algorithm without prediction to obtain another solution. Then, these two solutions are fed into a meta algorithm running Algorithm 2 to obtain the final solution. This obtains Theorem 6.1, which we stated in Section 6.