Online Primal-Dual Algorithm with Predictions for Non-Linear Covering Problems

Kevi Enikő
LIG, University Grenoble-Alpes, France

Nguyễn Kim Thắng
LIG, University Grenoble-Alpes, France

Abstract

Designing online algorithms with predictions is a recent technique for various practically relevant online problems (scheduling, caching (paging), clustering, ski rental, etc.). [8] provided a unified approach through a primal-dual framework for linear covering problems. Their framework extends the online primal-dual method by incorporating predictions, and its performance goes beyond the worst-case analysis. Following this research line, our paper presents competitive algorithms with predictions for *non-linear* covering problems, generalizing the previous technique. We illustrate the applicability of our algorithms through experiments on energy minimization, congestion management, and submodular minimization problems.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases online algorithms, predictions, primal-dual, non-linear objective functions, covering

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

Online computation (coined by [9]) is a well-established field in theoretical computer science. The performance of an online algorithm is typically measured in the worst-case paradigm by the competitive ratio metric, which is the worst ratio between the objective value obtained by the algorithm and that of the optimal solution. The traditional worst-case analysis is an indispensable framework in algorithm design and is central in the development of algorithms. Nevertheless, it can lead practical users to several pitfalls. Summarizing an algorithm's performance by a pathological worst-case can overestimate its performance on average.

Much of the research focused on going beyond the worst-case paradigm [31, 32] is motivated by the spectacular advances of machine learning (ML). Specifically, ML methods can detect patterns among the arriving input requests and provide valuable insights for the online algorithms regarding future requests. [25] introduced a general framework to integrate ML predictions into classical algorithm designs to surpass the worst-case performance limit.

Even though predictions provide a glimpse of the future, there is no mathematical guarantee for their accuracy. Adjusting the algorithm's trust in the predictions is a significant challenge since online algorithms must make irrevocable decisions at each time step. Ideally, if the predictions are accurate, the algorithm should perform well compared to the offline setting. In contrast, if the predictions are misleading, the algorithm should maintain a competitive solution, similar to the online setting where no predictive information is available. In other words, online algorithms with predictions are expected to bring the best of both worlds: mathematical performance guarantees of classical algorithms and good future prediction capabilities of machine learning methods.

To overcome the issue of unknown prediction accuracy, the authors of the works we cited previously exploited specific structures within the studied problems. [8] presented a primal-dual method based technique to unify these different ad-hoc approaches and design online algorithms with predictions for various online problems. The primal-dual method is an elegant and powerful algorithm design technique (introduced by [36]), especially for online algorithms (see [11]). The work of [8] focuses on problems with linear objectives and covering constraints. They raised an open question to design online algorithms with predictions for *non-linear* covering problems. Non-linear objectives appear naturally in diverse application domains, such as energy and congestion management. Therefore, answering this open question has high theoretical interest and vital practical motivations. Our paper presents a framework to design online primal-dual algorithms with predictions for covering problems with non-linear objectives.

1.1 Model

Building upon the work of [8] (which has several definitions rooted in [25, 23]), our model includes a prediction oracle $\mathcal{P}$ and a parameter $\eta \in (0, 1]$ which characterizes the confidence in the predictions. Small η values represent low doubt, meaning that the prediction accuracy is high, while large η values show high doubt, signaling that the predictions should be discarded. Given an online problem, upon the arrival of the current request, the online algorithm solving the problem must make an irrevocable decision regarding the request while satisfying the problem's constraints. In our setting, the decision-making is influenced by the prediction of the oracle $\mathcal{P}$, the confidence parameter η, the current solution, and the history of released requests. Intuitively, the oracle's predictions provide information about the unknown future. For example, it can predict the optimal machine for the current task during scheduling. To characterize the performance of an online algorithm with predictions, we use the notion of consistency and robustness. An algorithm $\mathcal{A}$ (for a minimization problem) is $C(\eta)$-*consistent* and $R(\eta)$-*robust* if for every instance I,

$$\mathcal{A}(I) \leq \min\{C(\eta) \cdot \mathcal{P}(I), R(\eta) \cdot \mathcal{O}(I)\}$$

where $\mathcal{A}(I), \mathcal{P}(I), \mathcal{O}(I)$ are respectively the objective values on instance I of algorithm $\mathcal{A}$, the prediction oracle $\mathcal{P}$ and the optimal offline solution $\mathcal{O}$. Following the convention, when the prediction oracle $\mathcal{P}$ provides an infeasible solution, $\mathcal{P}(I)$ is set to $+\infty$.Ideally, when η approaches 0 (high confidence in the prediction), $C(\eta)$ should tend to 1. Meanwhile, when η approaches 1 (high doubt in the prediction), $R(\eta)$ should tend towards the best competitive ratio in the classic online setting.

Similarly to the work of [8], our algorithm $\mathcal{A}$ combines the predictions of oracle $\mathcal{P}$ with the primal-dual method. This method formulates the studied problem as a mathematical program called the primal and its corresponding dual. Considering an online problem, at the arrival of a new request, a primal-dual method based online algorithm updates its fractional solutions to both the primal and dual programs to maintain feasibility (satisfy the constraints of the mathematical programs). The competitive ratio of such an algorithm is established by showing that every time the algorithm updates the primal and dual solutions, the increase of the primal objective value can be bounded by that of the dual up to some desired factor.

Our presented model contains two components by design. One relates to the prediction oracle, and the other to the classical primal-dual method. This duality is also present during the performance evaluation since our algorithms must achieve both good consistency and robustness. Given two separate algorithms, where one blindly follows the predictions while the other makes decisions solely based on the primal-dual method, a natural question is whether a simple linear combination of the two algorithms performs well. If we target a consistency of at least $O(1/(1-\eta))$, using a linear combination of the two algorithms, the robustness must be $\Omega(1/\eta)$. However, the ultimate goal is to achieve robustness in the order of poly($\log(1/\eta)$) (exponentially smaller than $\Omega(1/\eta)$) while maintaining $O(1/(1-\eta))$ consistency. Therefore, a simple linear combination of the two components is insufficient to reach the desired performance guarantees.

Our paper presents a framework for non-linear online covering problems with an intricate combination of the classic primal-dual method and a prediction oracle. Algorithms created with our framework construct fractional solutions, which is the primary step in primal-dual method based algorithms. Even though many real-life problems require integer solutions, online rounding schemes already exist for most of them. We provide references to such rounding schemes at the analysis of our studied problems.

1.2 Contribution

Inspired by the approach of [8], our model (detailed previously) combines an oracle's predictions with the primal-dual method in a way that the oracle's predictions influence the updates of the primal and dual variables. The construction of our algorithm follows the multiplicative weight update method based on the gradient of the multilinear extension of the problem's objective function ([34], see Section 2). This technique generalizes the multiplicative weight update introduced in [10, 3]. Using the local-smoothness notion of the multilinear extension, we can prove the feasibility of the primal and dual solutions (even when the prediction is infeasible). Afterwards, the algorithm's performance is established using the local-smoothness and confidence parameters.

▶ **Theorem 1.** *(Informal) Given a non-linear online covering problem, let F be the multilinear extension (see section 2 Preliminaries) of the problem's objective function. Assuming (λ,μ)-local-smoothness properties on F, for every confidence parameter η of the prediction oracle, where $\eta \in (0,1]$, there exists an $O\big(\frac{1}{1-\eta}\big)$-consistent and $O\big(\frac{\lambda}{1-\mu} \cdot \ln\left(\frac{d}{\eta}\right)\big)$-robust algorithm for the non-linear online fractional covering problem, where d is the maximum raw sparsity of the problem's constraints (maximum number of non-zero coefficients in a constraint).*

Example. We provide a concrete example for a better understanding/appreciation of Theorem 1 (which relies on several parameters). Consider the objective function as a polynomial of degree k with non-negative coefficients. For this class of functions, parameters $\lambda = \Theta(k^k)$ and $\mu = (k-1)/k$ [34]. The competitive ratio of state-of-the-art online algorithms *without* predictions is in $O(k^k \log d)$. In the setting with predictions, the consistency of our framework is $O(1/(1-\eta))$, and the robustness is $O(k^k \log(d/\eta))$.

1.3 Applications

We show the applicability of our framework through the following problems.

Load Balancing. Load balancing is a classic problem in discrete optimization with wide-ranging applications (for example, resource management in data centres). This problem revolves around assigning jobs that arrive online to m available unrelated machines while minimizing their maximum load. Our framework provides an $O(\frac{1}{1-\eta})$-consistent and and $O\big((\log m)\log^2 \frac{m}{\eta}\big)$-robust algorithm with fractional solution to this problem.

Energy Minimization in Scheduling. Reducing carbon emissions is a global effort in which energy-efficient algorithms play an essential role. For example, [1] and [18] studied energy-efficient algorithms for scheduling. Contrary to performance-oriented scheduling, our goal is to design an assignment policy of jobs to m available machines, which can minimize the total energy consumption of the execution. Energy-related objective functions are typically polynomials of degree $k > 1$. Using our proposed framework, we can derive an $O(\frac{1}{1-\eta})$-consistent and $O\left(k^k \log^k \frac{m}{\eta}\right)$-robust algorithm with fractional solutions for this energy minimization problem.

Submodular Minimization. Submodular minimization is a widespread subject in optimization and machine learning ([20, 5, 4, 7]). We present a $O(\frac{1}{1-\eta})$-consistent and $O\left(\frac{\log(d/\eta)}{1-\kappa_f}\right)$-robust algorithm for minimizing a submodular function under covering constraints where κ_f is the curvature (defined in Section 3.1.3) of the submodular function.

1.4 Experiments

The experiments focus on a high-impact congestion management problem: network and transportation routing. The input is a directed graph $G(A, V)$ and a set of requests $R = \{(s_i, t_i) : s_i, t_i \in V\}$ that represents demands (connecting s_i to t_i through a path). Each arc $(u, v) \in A$ is associated with a cost function $f_{(u,v)} : \mathbb{R}^+ \to \mathbb{R}^+$ that depends on the number of requests using the arc. The goal is to design a routing that minimizes the total cost while requests arrive online. We enable predictions by building an oracle using the observed data. The oracle provides traffic forecasts, vital information to improve the routing. The experiments show that our algorithm outperforms both the best theoretical algorithm and the prediction in practical settings.

1.5 Related work

The primal-dual method is a powerful tool in online optimization. [10] introduced a primal-dual framework for linear programs with packing and covering constraints. Their method unifies several previous potential-function-based analyses and gives a principled approach to the design and analysis of algorithms for problems with linear relaxations. [3] provided a framework for covering and packing problems with convex and concave objectives whose gradients are monotone. Subsequently, [34] presented algorithms without the convex assumption on the objective function and established a competitive ratio parameterized by the function's smoothness properties. This smoothness notion of [34] has roots in smooth games, which [30] defined in the context of algorithmic game theory.

The domain of algorithms with predictions ([28]) - or learning augmented algorithms - emerged recently and grown immensely at the intersection of (discrete) algorithm design and machine learning (ML). Combining ML techniques with traditional algorithm design methods enables online algorithms to benefit from predictions that can infer future information from patterns in past data. Online algorithms with predictions can obtain performance guarantees beyond the worst-case analysis and provide fine-tuned solutions to various problems. In the literature, many significant problems have new learning-augmented results. For example, scheduling ([24, 27]), caching (paging) ([25, 29, 2]), ski rental ([16, 23]), counting sketches ([19]), and bloom filters ([22, 26]). To design online algorithms with predictions in a unified way, [8] proposed a primal-dual approach for online linear problems with covering constraints. Since then, [17] further generalized this method for online *semidefinite* programming with covering constraints. By combining their ideas and the ones in [10, 3, 34], we present a primal-dual framework for general problems with *non-linear* objectives and covering constraints. Hence, this paper answers an open question in [8].

2 Preliminaries

Multilinear extension. We follow the primal-dual approach of [34] to design competitive algorithms for online fractional non-linear covering problems. This method uses the objective function's *multilinear extension* to characterize how far the objective function is from being linear. Given a function $f:\{0,1\}^n \to \mathbb{R}^+$, its multilinear extension $F:[0,1]^n \to \mathbb{R}^+$ is defined as $F(\boldsymbol{x}) := \sum_S \prod_{e\in S} x_e \prod_{e\notin S}(1-x_e)\cdot f(\mathbf{1}_S)$ where $\mathbf{1}_S$ is the characteristic vector of S (the e^{th}-component of $\mathbf{1}_S$ equals 1 if $e \in S$ and equals 0 otherwise). Alternatively, $F(\boldsymbol{x}) = \mathbb{E}\big[f(\mathbf{1}_T)\big]$ where T is a random set such that a resource e appears in T independently with probability x_e. We highlight that $F(\mathbf{1}_S) = f(\mathbf{1}_S)$ and define the following crucial property.

▶ **Definition 2** ([34]). *Let $\mathcal{E}$ be a set of n resources. A differentiable function $F:[0,1]^n \to \mathbb{R}^+$ is* (λ,μ)-locally-smooth *if for every set $S \subseteq \mathcal{E}$, and for every set of $|S|$ arbitrary vectors $\boldsymbol{x}^e \in [0,1]^n$ where $e \in S$, the following inequality holds:*

$$\sum_{e\in S} \nabla_e F(\boldsymbol{x}^e) \leq \lambda F\big(\mathbf{1}_S\big) + \mu F\big(\boldsymbol{x}\big)$$

where $\boldsymbol{x}$ is a vector whose every coordinate $x_{e'} = \max_e\{x^e_{e'}\}$ (formally, $\boldsymbol{x} := \bigvee_{e\in S} \boldsymbol{x}^e$); and $\nabla_e F(\boldsymbol{x})$ denotes $\partial F(\boldsymbol{x})/\partial x_e$.

The defined (λ,μ)-smoothness property differs from the standard notion of function smoothness used in convex optimization. Following the definition of [34], the current (λ,μ)-smoothness property relates to the definition of smooth games in the context of algorithmic game theory ([30]). Intuitively, given a (λ,μ)-locally-smooth function, the quantity $\frac{\lambda}{1-\mu}$ measures how far the function is from being linear. If a function is linear, it is $(1,0)$-locally smooth.

Definition 2 addresses general functions with non-monotone gradients. When $\nabla_e F(\boldsymbol{x})$ is non-decreasing on every coordinate e, we can simplify the definition.

▶ **Definition 3** ([34]). ***(+ monotone gradient assumption)*** *Let $\mathcal{E}$ be a set of n resources. A differentiable function $F:[0,1]^n \to \mathbb{R}^+$ with monotone gradient is* (λ,μ)-locally-smooth *if for every set $S \subseteq \mathcal{E}$, and for every arbitrary vector $\boldsymbol{x} \in [0,1]^n$, the following inequality holds:*

$$\sum_{e\in S} \nabla_e F(\boldsymbol{x}) \leq \lambda F\big(\mathbf{1}_S\big) + \mu F\big(\boldsymbol{x}\big)$$

Covering Problem. Let $\mathcal{E}$ be a set of n resources and let $f:\{0,1\}^n \to \mathbb{R}^+$ be an *arbitrary* non-decreasing function defined on the set. Let $x_e \in \{0,1\}$ be a variable indicating whether resource e is selected. In contrast to packing problems, the set of resources is known in advance, but the covering constraints $(\sum_e a^t_e x_e \geq 1)$ are revealed one-by-one over time. When a constraint arrives, the oracle gives a prediction $(pred(x_e) \in \{0,1\})$ for each resource e and our algorithm updates the solution $\boldsymbol{x} = (x_e)_{e\in\mathcal{E}}$ by only increasing the x_e variables. Online algorithms must make irrevocable decisions, which means that they cannot decrease the value of the decision variables. The update must always satisfy every revealed constraint. The objective is to minimize $f(\boldsymbol{x})$ subject to the online covering constraints.

3 Primal-Dual Framework for Covering Constraints

Formulation. We formulate the online covering problem that we described in the Preliminaries as a problem of finding the minimum cost solution among all the possible solutions.

This formulation has an exponential number of variables and constraints; however, it allows us to transform the non-linear objective function into a linear one, which is crucial for our algorithm and proofs.

Let $S \subseteq \mathcal{E}$ be a *solution* if $\mathbf{1}_S$ corresponds to a feasible solution. Let x_e be a variable indicating whether resource e is selected. Let z_S be an indicator variable for solution S. If $z_S = 1$, then every variable $x_e = 1$ if $e \in S$, and $x_e = 0$ if $e \notin S$. Otherwise, $z_S = 0$. In other words, $z_S = 1$ if and only if $\mathbf{1}_S$ is the selected solution of the online covering problem. At each time step t during the execution, a new constraint is revealed. For every subset $A \subseteq \mathcal{E}$, we define the value $c^t(A) := \max\{0,\ 1 - \sum_{e \in A} a_e^t\}$, to be the amount we need until constraint satisfaction. Given this value, we normalize the constraint coefficients to be $a_e^t(A) := \min\{a_e^t,\ c^t(A)\}$. Finally, we define $b_e^t(A) := a_e^t(A)\ /\ c^t(A)$ where $c^t(A) > 0$. The values $b_e^t(A)$ correspond to the coefficients in the knapsack inequality constraints ([13]). The primal and dual programs are:

$$
\begin{aligned}
\min \sum_{S \subseteq \mathcal{E}} f(\mathbf{1}_S)\, z_S & \\
\sum_{e \notin A} b_e^t(A)\, x_e \geq 1 \qquad & \forall t,\ \forall A \subseteq \mathcal{E} \\
\sum_{S : e \in S} z_S = x_e \qquad & \forall e \\
\sum_{S \subseteq \mathcal{E}} z_S = 1 \qquad & \\
x_e, z_S \in \{0,1\} \qquad & \forall e,\ \forall S \subseteq \mathcal{E}
\end{aligned}
\qquad\qquad
\begin{aligned}
\max \sum_{t,A} \alpha_A^t + \gamma & \\
\sum_t \sum_{A : e \notin A} b_e^t(A)\, \alpha_A^t \leq \beta_e \qquad & \forall e \\
\gamma + \sum_{e \in S} \beta_e \leq f(\mathbf{1}_S) \qquad & \forall S \subseteq \mathcal{E} \\
\alpha_A^t \geq 0 \qquad & \forall t,\ \forall A \subseteq \mathcal{E} \\
\beta_e \geq 0 \qquad & \forall e \\
\gamma \geq 0 \qquad &
\end{aligned}
$$

In the primal program, the first constraints are knapsack-constraints ([13]) of the given polytope, and they are equivalent to $\sum_{e \notin A} a_e^t(A)\ x_e \geq c^t(A)$. It is sufficient to satisfy constraints where $c^t(A) > 0$. The second primal constrain ensures that if a resource e is chosen, the selected solution must contain e. The third constraint guarantees that *one* solution is selected.

Algorithm. In our proposed algorithm, $\boldsymbol{x} \in [0,1]^{|\mathcal{E}|}$ corresponds to the current solution of the algorithm. During the execution, we rely on the objective function's multilinear extension F, parametrized by λ and μ. We assume, that $F(\boldsymbol{x})$ is $(\lambda, C\mu)$-locally-smooth, where C is a constant that arises from the algorithm's analysis (see Lemma 5). Algorithm 1 follows the scheme of [34], which uses both the primal and dual variables to solve the problem.

We have two notions of time in our algorithm. First, at each discrete time step t, a new primal constraint arrives. Second, we have a continuous time τ throughout the execution. The solution of the algorithm increases gradually with time τ. To simplify the notations, when the context only uses the current time of the execution, $\boldsymbol{x}$ refers to $\boldsymbol{x}(\tau)$, the current solution at time τ.

When a new primal constraint arrives, the current dual variable $\alpha_{A^*}^t$ increases at a constant rate (line 8), while the β_e variables are updated according to the partial derivative of the mulitlinear extension (line 10). We note a subtle point here: if $\beta_e < \frac{1}{\lambda}\nabla_e F(\boldsymbol{x})$ then we set $\beta_e = \frac{1}{\lambda}\nabla_e F(\boldsymbol{x})$, but if $\beta_e > \frac{1}{\lambda}\nabla_e F(\boldsymbol{x})$ then we do not change the value of β_e. This update preserves the following invariants during the execution of the algorithm: $\beta_e \geq \frac{1}{\lambda}\nabla_e F(\boldsymbol{x})$ and β_e is non-decreasing. (Remark: if $\nabla_e F(\boldsymbol{x})$ is monotone on every coordinate e, then it is sufficient to always set $\beta_e \leftarrow \frac{1}{\lambda}\nabla_e F(\boldsymbol{x})$.)

Algorithm 1 Online Algorithm for Non-Linear Covering Problems.

1: Initially, set $A^* \leftarrow \emptyset$ # (where A^* is the solution set and $\forall e \in A^* : x_e = 1$)
2: All primal and dual variables are initially set to 0
3: During every step, for each feasible solution S, $z_S = \prod_{e \in S} x_e \prod_{e \notin S}(1 - x_e)$ is maintained.
4: Let τ be the continuous timer during the execution of the algorithm.
5: **for** each time t, for the new primal constraint $\sum_e a_e^t x_e \geq 1$ and dual variable $\alpha_{A^*}^t$ **do**
6: **while** $\sum_{e \notin A^*} b_e^t(A^*)\, x_e < 1$ **do** # Increase primal, dual variables
7: Increase τ with a rate of 1.
8: Increase $\alpha_{A^*}^t$ at rate 1 / $(\lambda \, \ln(1 + 2d^2/\eta))$
9: **for** $e \notin A^*$ such that $b_e^t(A^*) > 0$ **do**
10: **if** $\beta_e < \frac{1}{\lambda} \nabla_e F(\boldsymbol{x})$ **then** $\beta_e \leftarrow \frac{1}{\lambda} \nabla_e F(\boldsymbol{x})$
11: Increase x_e at a rate according to the following

$$\frac{\partial x_e}{\partial \tau} \leftarrow \frac{b_e^t(A^*)\, x_e}{\lambda \beta_e} + \frac{\eta}{\lambda \beta_e d} + \frac{(1-\eta) \cdot \mathbb{1}_{\{pred(x_e)=1\}}}{\nabla_e F(\boldsymbol{x}) \cdot |\{e' : pred(x_{e'}) = 1,\ b_{e'}^t(A^*) > 0\}|}$$

12: **end for**
13: **if** $x_e = 1$ **then** $A^* \leftarrow A^* \cup \{e\}$
14: **for** $e : e \notin A^*$ **do** # Decrease dual variables
15: **while** $\sum_{t'=1}^{t} \sum_{A: e \notin A} b_e^{t'}(A)\, \alpha_A^{t'} > \beta_e$ **do**
16: **for** (t_e^*, A) such that $b_e^{t_e^*}(A) = \max\{b_e^{t'}(A) \mid \forall A : e \notin A$ and $\forall t' \leq t$ s.t. $\alpha_A^{t'} > 0\}$ **do**
17: Decrease $\alpha_A^{t_e^*}$ continuously with a rate of $\frac{b_e^t(A^*)}{b_e^{t_e^*}(A)} \cdot \frac{1}{\lambda \cdot \ln(1+2d^2/\eta)}$
18: **end for**
19: **end while**
20: **end for**
21: **end while**
22: **end for**

The update on line 11 is inspired by the multiplicative weight update method (where the increasing rate of x_e is inversely proportional to β_e) and the updating approach of [8]. Starting from line 14, the algorithm decreases some of the dual variables using a similar idea as in [3]. This decrease is necessary to maintain the feasibility of the dual solution.

Primal and dual variables. Let $\boldsymbol{x}(\tau)$ be the algorithm's primal solution at time τ. The dual variables α_A^t and β_e are assigned during the execution, but not γ. To make the dual solution feasible, we set $\gamma = -\frac{\mu}{4\lambda \cdot \ln(1+2d^2/\eta)} F(\boldsymbol{x}(\tau))$ (see Lemma 5). Each $\beta_e = \frac{1}{\lambda} \nabla_e F(\boldsymbol{x}(\tau'))$, for some primal solution $\boldsymbol{x}(\tau')$, where $\tau' \leq \tau$. Moreover, $\boldsymbol{x}(\tau) \geq \bigvee_{\tau' \leq \tau} \boldsymbol{x}(\tau')$ (each coordinate $x_e(\tau) = \max_{\tau' \leq \tau}\{x_e(\tau')\}$), since the x_e-variables are non-decreasing. Consequently, each $\beta_e \geq \frac{1}{\lambda} \nabla_e F(\boldsymbol{x}(\tau))$. Using these properties, Lemma 4 gives a lower bound on $\boldsymbol{x}(\tau)$. We highlight that the proof if this lemma does *not* require the gradient of F to be monotone. (When this assumption is present, the algorithm can simply set $\beta_e = \frac{1}{\lambda} \nabla_e F(\boldsymbol{x}(\tau))$ at each step τ of the execution.)

▶ **Lemma 4.** *Let e be an arbitrary resource. At any moment τ during the execution of the* *algorithm, when t constraints have already been released, it always holds that*

$$x_e \geq \frac{\eta}{b_e^{t_e^*}(A) \cdot d} \left[\exp\left(\frac{\ln(1+2d^2/\eta)}{\beta_e} \cdot \sum_{A: e \notin A} \sum_{t' \leq t} b_e^{t'}(A) \cdot \alpha_A^{t'} \right) - 1 \right]$$

(where recall that $b_e^{t_e^*}(A)$ is defined in the algorithm on line 16).

▶ **Lemma 5.** *The primal and dual variables are feasible.*

Proof. Primal feasibility. While a primal covering constraint is unsatisfied, the x_e-variables are increasing. At the end of the first iteration, the first primal covering constraint is satisfied. Afterwards, the new constraints are also satisfied, since the algorithms maintains $z_S = \prod_{e \in S} x_e \prod_{e \notin S}(1-x_e)$. If we choose an element e with probability x_e, then z_S is the probability that the set of selected items is S. Therefore, the total probability $\sum_S z_S = 1$. By a similar argument, we get the following: $\sum_{S: e \in S} z_S = x_e \sum_{S' \subseteq E \setminus \{e\}} \prod_{e' \in S'} x_{e'} \prod_{e' \notin S'}(1 - x_{e'}) = x_e$ since $\sum_{S' \subseteq E \setminus \{e\}} \prod_{e' \in S'} x_{e'} \prod_{e' \notin S'}(1 - x_{e'}) = 1$.

Dual feasibility. Let us now prove that the first dual constraint is always satisfied during the execution. The algorithm maintains $\sum_{t' \leq t} \sum_{A: e \notin A} b_e^{t'}(A)\ \alpha_A^{t'} \leq \beta_e$. Whenever this inequality is violated, the algorithm decreases (see line 17) some of the α-variables in a way that the increasing rate of $\sum_{t' \leq t} \sum_{A: e \notin A} b_e^{t'}(A)\ \alpha_A^{t'}$ is at most 0. By the β-variables' definition, the first dual constraint holds.

Let us consider the second dual constraint. Let $\boldsymbol{x}(\tau)$ be the final solution of the algorithm. For each fixed resource e, the value $\beta_e = \frac{1}{\lambda} \nabla_e F(\boldsymbol{x}(\tau_e))$ for some previous solution $\boldsymbol{x}(\tau_e)$ where $\tau_e \leq \tau$ and where $x_e(\tau_e) \leq x_e(\tau)$ for all e. Let $\boldsymbol{y} := \bigvee_{\tau' \leq \tau} \boldsymbol{x}(\tau') \leq \boldsymbol{x}(\tau)$, so for each coordinate e of $\boldsymbol{y}$, we have $y_e = \max_{\tau' \leq \tau}\{x_e(\tau')\}$. By definition of the dual variables, the second dual constraint (after rearranging the terms) reads

$$\frac{1}{\lambda} \sum_{e \in S} \nabla_e F(\boldsymbol{x}(\tau_e)) \leq F(\mathbf{1}_S) + \frac{\mu}{4\lambda \cdot \ln(1+2d^2/\eta)} F(\boldsymbol{x}(\tau)) \qquad \forall\ S \subseteq \mathcal{E}$$

since we set $\gamma = -\frac{\mu}{4\lambda \cdot \ln(1+2d^2/\eta)} F(\boldsymbol{x}(\tau))$, and $\boldsymbol{x}(\tau_e)$ corresponds to the solution during the execution where β_e was set to $\frac{1}{\lambda} \nabla_e F(\boldsymbol{x}(\tau_e))$. Since F is monotone, $F(\boldsymbol{x}(\tau)) \geq F(\boldsymbol{y})$. To prove that the above inequality holds, it is sufficient to show that

$$\frac{1}{\lambda} \sum_{e \in S} \nabla_e F(\boldsymbol{x}(\tau_e)) \leq F(\mathbf{1}_S) + \frac{\mu}{4\lambda \cdot \ln(1+2d^2/\eta)} F(\boldsymbol{y})$$

which means that F needs to be $\big(\lambda, \frac{\mu}{4\ln(1+2d^2/\eta)}\big)$-locally-smooth. Our initial assumption was that F is $(\lambda, C\mu)$-locally-smooth. By setting $C := \frac{1}{4\ln(1+2d^2/\eta)}$, the lemma holds. ◀

▶ **Theorem 1.** *Let F be the multilinear extension of the online non-linear covering problem's objective function f and d be the maximal row sparsity of the constraint matrix (formally $d = \max_{t \leq T} |\{a_e^t : a_e^t > 0\}|$). Assuming that F is $\big(\lambda, \frac{\mu}{\ln(1+2d^2/\eta)}\big)$-locally-smooth for some parameters ($\lambda > 0$, $\mu < 1$ and $0 < \eta \leq 1$), there exists an $O\big(\frac{1}{1-\eta}\big)$-consistent and $O\big(\frac{\lambda}{1-\mu} \cdot \ln \frac{d}{\eta}\big)$-robust algorithm for any $\eta \in (0,1]$ which produces a fractional solution for the given problem.*

Proof. Robustness. By bounding the increases of the primal and dual objective values at any time τ during the execution of Algorithm 1, we can determine the robustness. Upon the release of the t^{th} constraint, let A^* be the current solution with the chosen set of resources e such that $x_e = 1$.

The derivative of the primal objective function with respect to τ is:

$$\sum_{e \in \mathcal{E}} \nabla_e F(\boldsymbol{x}) \cdot \frac{\partial x_e}{\partial \tau}$$
$$= \sum_{e: b_e^t(A^*)>0} \nabla_e F(\boldsymbol{x}) \left(\frac{b_e^t(A^*) x_e}{\lambda \beta_e} + \frac{\eta}{\lambda \beta_e\ d} + \frac{(1-\eta)\mathbb{1}_{\{pred(x_e)=1\}}}{\nabla_e F(\boldsymbol{x}) \cdot |\{e' : pred(x_{e'}) = 1, b_{e'}^t(A^*) > 0\}|} \right)$$
$$\leq \sum_{e: b_e^t(A^*)>0} \left(b_e^t(A^*)\ x_e + \frac{\eta}{d} \right) + \sum_{\substack{e: pred(x_e)=1 \\ b_e^t(A^*)>0}} \frac{(1-\eta)}{|\{e' : pred(x_{e'}) = 1,\ b_{e'}^t(A^*) > 0\}|} \leq 2 \qquad (1)$$

The first inequality follows $\nabla_e F(\boldsymbol{x}) \leq \lambda\ \beta_e$. The second inequality is due to the definition of d and the fact that $\sum_{e \notin A^*} b_e^t(A^*)\ x_e \leq 1$ always holds during the algorithm. (The number of $b_e^t(A^*)$ values which are strictly greater than 0, is at most d.)

At any time τ, let $U(\tau)$ be the set of resources e such that $\sum_{t' \leq t} \sum_{A: e \notin A} b_e^{t'}(A)\ \alpha_A^{t'} = \beta_e$ and $b_e^t(A^*) > 0$. Note that $|U(\tau)| \leq d$ by definition of d. As long as $\sum_{e \notin A^*} b_e^t(A^*)\ x_e < 1$, using Lemma 4 we get that for every $e \in U(\tau)$,

$$\frac{1}{b_e^t(A^*)} > x_e \geq \frac{\eta}{b_e^{t_e^*}(A)\ d} \left[\exp\Big(\ln(1 + 2d^2/\eta) \Big) - 1 \right] = \frac{2d}{b_e^{t_e^*}(A)}$$

where $b_e^{t_e^*}(A)$ is defined in the algorithm on line 16. Therefore, $\frac{b_e^t(A^*)}{b_e^{t_e^*}(A)} \leq \frac{1}{2d}$.

Following the definition of $U(\tau)$, we can bound the increase of the dual at time τ. The derivative of the dual with respect to t is:

$$\frac{\partial D}{\partial \tau} = \sum_{t' \leq t} \sum_{A: e \notin A} \frac{\partial \alpha_A^{t'}}{\partial \tau} + \frac{\partial \gamma}{\partial \tau} = \sum_{t' \leq t} c^{t'}(A^*) \cdot \frac{\partial \alpha_{A^*}^{t'}}{\partial \tau} + \frac{\partial \gamma}{\partial \tau}$$
$$= \frac{1}{\lambda \cdot \ln(1 + 2d^2/\eta)} \cdot \left(1 - \sum_{e \in U(\tau)} \frac{b_e^t(A^*)}{b_e^{t_e^*}(A)} \right) - \frac{\mu}{4\lambda \cdot \ln(1 + 2d^2/\eta)} \cdot \sum_e \nabla_e F(\boldsymbol{x}) \frac{\partial x_e}{\partial \tau}$$
$$\geq \frac{1}{\lambda \cdot \ln(1 + 2d^2/\eta)} \left(1 - \sum_{e \in U(\tau)} \frac{1}{2d} \right) - \frac{\mu}{2\lambda \cdot \ln(1 + 2d^2/\eta)}$$
$$\geq \frac{1-\mu}{2\lambda \cdot \ln(1 + 2d^2/\eta)}.$$

The third equality holds since $\alpha_{A^*}^t$ is increased and other α-variables in $U(\tau)$ are decreased. The first inequality uses the fact that $\frac{b_e^t(A^*)}{b_e^{t_e^*}(A)} \leq \frac{1}{2d}$ and Inequality (1). The last inequality holds since $|U(\tau)| \leq d$. Hence, the robustness is at least $\frac{4\lambda}{1-\mu} \cdot \ln(1 + 2d^2/\eta)$.

Consistency. We establish consistency with a similar argument as [8]. Considering an arbitrary moment τ during the algorithm's execution, let $S_1 = S_1(\tau)$ be the set of resources selected by the prediction. Formally, $\forall\ e \in S_1 : pred(x_e) = 1$ up to time τ. Let $S_2 = S_2(\tau)$ contain the remaining resources. The primal objective increases due to S_1 and S_2 are:

$$\sum_{e \in S_1} \nabla_e F(\boldsymbol{x})\ \frac{\partial x_e}{\partial \tau} = \sum_{e \in S_1} \nabla_e F(\boldsymbol{x})\ \left(\frac{b_e^t(A^*)\ x_e}{\lambda\ \beta_e} + \frac{\eta}{\lambda\ \beta_e\ d} + \frac{(1-\eta)}{\nabla_e F(\boldsymbol{x}) \cdot |\{e' : pred(x_{e'}) = 1\}|} \right)$$
$$\geq 1 - \eta$$
$$\sum_{e \in S_2} \nabla_e F(\boldsymbol{x})\ \frac{\partial x_e}{\partial \tau} = \sum_{e \in S_2} \nabla_e F(\boldsymbol{x})\ \left(\frac{b_e^t(A^*)\ x_e}{\lambda\ \beta_e} + \frac{\eta}{\lambda\ \beta_e\ d} \right) \leq 1 + \eta$$

Therefore, the primal objective increase is at most $\left(1 + \frac{1+\eta}{1-\eta}\right)$ time the increase restricted to the set S_1. Moreover, the algorithm's primal objective value restricted to S_1 is smaller than the prediction's, since $\forall e \in S_1 : x_e \leq 1 = \mathit{pred}(x_e)$. We can deduce that the algorithm is $O\left(\frac{1}{1-\eta}\right)$-consistent with the prediction. ◀

3.1 Applications

To apply Theorem 1 on specific problems, we need to determine the local-smoothness parameters for the multilinear extension. [34] provided these parameters for some broad classes of functions, in particular for polynomials with non-negative coefficients. Let $g_\ell : \mathbb{R} \to \mathbb{R}$ for $1 \leq \ell \leq L$ be degree-k polynomials with non-negative coefficients and let $f : \{0,1\}^n \to \mathbb{R}^+$ be the cost function defined as $f(\mathbf{1}_S) = \sum_\ell b_\ell g_\ell\left(\sum_{e \in S} a_e\right)$ where $a_e \geq 0$ for every e and $b_\ell \geq 0$ for every $1 \leq \ell \leq L$. Then the multilinear extension F of f is $\left(O(k \ln(d/\eta))^{k-1}, \frac{k-1}{k \ln(1+2d^2/\eta)}\right)$-locally smooth. We will use these parameters to derive the guarantees for the following problems.

3.1.1 Load Balancing

Problem. Load balancing is a classic problem in discrete optimization with wide-ranging applications (for example, resource management in data centres). This problem revolves around assigning jobs that arrive online to m available unrelated machines while minimizing their maximum load. Each arriving job j reveals its machine dependent execution time p_{ij} where $i \in \{1, m\}$ is the machine's index. The load ℓ_i of machine i is the total processing time of the jobs assigned to it. This load balancing problem is a well understood standard online problem and it has a tight competitive ratio of $\Theta(\log m)$ ([9, 12]).

In our online setting with predictions, the jobs not only arrive with their machine dependent execution time p_{ij}, but their machine dependent prediction as well. Formally, $x_{ij} \in \{0,1\}$ indicates whether job j is assigned to machine i, and the oracle provides $\mathit{pred}(x_{ij}) \in \{0,1\}$. We can formulate the online load balancing problem as a non-linear program. The objective is $\min \max_{i=1}^m \ell_i = \min \max_{i=1}^m \left(\sum_j p_{ij} x_{ij}\right)$, and the constraint is $\sum_{i=1}^m x_{ij} = 1$ which guarantees that each job j is assigned to some machine i. Applying our framework for non-linear programs with covering constraints, Proposition 6 follows.

▶ **Proposition 6.** *Algorithm 1 gives a $O(\frac{1}{1-\eta})$-consistent and $O\left((\log m) \log^2 \frac{m}{\eta}\right)$-robust fractional solution for the load balancing problem.*

3.1.2 Energy Minimization in Scheduling

Problem. Reducing carbon emissions is a global effort in which energy-efficient algorithms play an essential role. For example, [1] and [18] studied energy-efficient algorithms for scheduling.

Given m unrelated machines, we need to assign jobs that arrive online. Each job j has a release date r_j, a deadline d_j, and a vector of machine dependent processing times p_{ij}. Contrary to performance-oriented scheduling, our goal is to design an assignment policy which can minimize the total energy consumption of the execution. To achieve this, we can adjust the machines' speed $s_{ij}(t)$ during the time interval $[t, t+1)$ for the execution of job j. Every machine i has a non-decreasing energy power function $P_i(\cdot)$. Typically, $P_i(z) = z^{k_i}$ for some constant $k_i \geq 1$. The execution's total energy is $\sum_i \sum_t P(\sum_j s_{ij}(t))$.

In the classic online setting, this problem is well understood: there exists an $O(k^k)$-competitive algorithm ([34]) where $k = \max_i\{k_i\}$ and this bound is tight up to a constant factor ([12]). In our extended study with predictions we represent this problem with the following non-linear program. The objective is $\min \sum_i \sum_t P(\sum_j s_{ij}(t))$ and the constraints are:

$$\sum_{i=1}^{m} x_{ij} = 1, \qquad \sum_{t=r_j}^{d_j-1} s_{ij}(t) \geq p_{ij}x_{ij}, \qquad s_{ij}(t) \geq 0 \qquad \forall\, i,\ t$$

where $x_{ij} \in \{0,1\}$ indicates whether job j is assigned to machine i and $s_{ij}(t) \geq 0$ denotes the speed of machine i executing job j during the time interval $[t, t+1)$. The first constraint guarantees that job j is assigned to some machine, and the second one ensures that the job j is completed on time (on the machine where the job is assigned). At the arrival of job j, the prediction provides a solution $pred(x_{ij})$ and a speed $pred(s_{ij}(t))$ for $r_j \leq t \leq d_j - 1$. Using our framework, we can deduce the following result.

▶ **Proposition 7.** *Algorithm 1 gives a $O(\frac{1}{1-\eta})$-consistent and $O\big(k^k \log^k \frac{m}{\eta}\big)$-robust fractional solution for the energy minimization problem.*

3.1.3 Online Submodular Mimimization

Problem. Submodular minimization is a widespread subject in optimization and machine learning ([20, 5, 4, 7]). Let us consider the problem of minimizing an online monotone submodular function subject to covering constraints. A set-function $f : 2^{\mathcal{E}} \to \mathbb{R}+$ is *submodular* if $f(S \cup e) - f(S) \geq f(T \cup e) - f(T)$ for all $S \subset T \subseteq \mathcal{E}$. Let F be the multilinear extension of a monotone submodular function f. Function F admits two useful properties. First, if f is monotone, then so is F. Second, F is concave in the positive direction, meaning that $\nabla F(\boldsymbol{x}) \geq \nabla F(\boldsymbol{y})$ for all $\boldsymbol{x} \leq \boldsymbol{y}$, where $\boldsymbol{x} \leq \boldsymbol{y}$ is defined as $x_e \leq y_e$ $\forall e$.

To apply Algorithm 1, we need to determine the local-smoothness parameters. An important concept in studying submodular functions is the *curvature*. Given a submodular function f, the *total curvature* κ_f ([14]) of f is defined as $\kappa_f = 1 - \min_e \frac{f(\mathbf{1}_{\mathcal{E}}) - f(\mathbf{1}_{\mathcal{E}\setminus\{e\}})}{f(\mathbf{1}_{\{e\}})}$. Intuitively, the total curvature measures how far away f is from being *modular*. This concept of curvature is used to determine both upper and lower bounds on the approximation ratios for many submodular and learning problems (see [14, 15, 6, 35, 21, 33]).

▶ **Proposition 8.** *Algorithm 1 gives a $O(\frac{1}{1-\eta})$-consistent and $O\big(\frac{\log(d/\eta)}{1-\kappa_f}\big)$-robust fractional solution for the submodular minimization under covering constraints.*

4 Experiments

Setting. To evaluate the empirical performance of our proposed algorithm, we conducted experiments on routing problems that are motivated by congestion management. In the routing problem, we are given a directed graph $G(A, V)$, a set of requests $R = \{(s_i, t_i) : s_i, t_i \in V\}$ that represents demands of connecting s_i to t_i. We assume that for each request, there exists a directed path between s_i to t_i. Each arc $(u, v) \in A$ is associated with a cost function $f_{(u,v)} : \mathbb{R}^+ \to \mathbb{R}^+$ that depends on the number of requests using the arc. Requests arrive online, and one needs to design a routing that minimizes the total cost.

Input. We generate the input graphs randomly following the Erdős-Rényi model $G(n, p)$, where n is the number of vertices and p is the probability that an arc gets created. The source and target vertices of the requests are also generated uniformly at random.

Predictions. The predictions rely on the optimal offline integral solution. We define the error of a prediction P on instance I as $error(P(I)) = 1 - \frac{OPT(I)}{P(I)}$, where $OPT(I)$ is the objective value of the optimal offline integral solution of instance I and $P(I)$ is the objective value obtained by the prediction's solution. To introduce errors in the prediction, we choose a request uniformly at random and attempt to find an alternative path compared to the optimal integral solution. We repeat this process several times to raise the error rate above the desired threshold.

Implementation. The covering formulation of the routing problem enumerates all possible cuts in the graph. At each arriving request $r = (s, t)$, our algorithm receives a set of constraints, $\sum_{e \in \delta(S)} x_e^r \geq 1$, where $\delta(S)$ is the cut on $S \subset V$, such that $s \in S$ and $t \notin S$. Upon each arrival request, our algorithm receives two solutions: one from the prediction and one from a greedy algorithm. The greedy algorithm calculates, at each arriving request, a path that minimizes the increase of the total cost and routes the request on this path. It is shown that this routing has the optimal competitive ratio when the cost functions are polynomial [34, Section 4.2]. In our implementation, we update the arcs as described in Algorithm 1. The request is satisfied when a path exists among the arcs in the set A^* (arcs with value 1) in our algorithm. If such a path exists, the solution of the request is this path.

Instances. We show the results of several instances in Figure 1 and other figures in Appendix C. The graph of the experiment contains 40 vertices, 126 arcs, and 17 requests. The cost functions of the arcs are polynomials of degree 4, and the coefficients were generated randomly in $[1.0, 10.0]$.

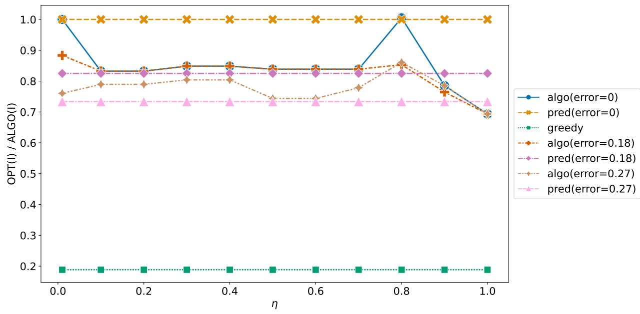

Figure 1 The x-axis shows the confidence in the prediction, where 0 means higher confidence. The y-axis shows the competitive ratio compared to the optimal offline integral solution. The different colors (also markers) show the result of the algorithm with different prediction error rates and the solutions of the greedy algorithm and the prediction alone.

Observation. When the prediction is not an optimal offline solution (the error is not 0), our algorithm outperforms both the prediction and the greedy solution when the confidence parameter is $0.1 \leq \eta \leq 0.8$. In practice, predictions are neither perfect nor completely wrong. So the confidence parameter, reflecting the reliability of the predictions, is rarely very close to 0 nor very close to 1. The experiments prove that our algorithm provides improvements over the predictions and the greedy solution (achieving the best theoretical performance) in the practical aspect of the routing problem.

5 Conclusion

We presented a primal-dual framework to design algorithms with predictions for non-linear covering problems. The framework provides useful ideas to incorporate predictions into algorithms with applications on widely-studied problems, such as load balancing, energy minimization, and submodular minimization. An interesting research direction is designing algorithms for non-linear packing problems and in the setting of multiple predictions.

References

1 Susanne Albers. Energy-efficient algorithms. *Commun. ACM*, 53(5):86–96, 2010.

2 Antonios Antoniadis, Christian Coester, Marek Elias, Adam Polak, and Bertrand Simon. Online metric algorithms with untrusted predictions. In *International Conference on Machine Learning*, pages 345–355, 2020.

3 Yossi Azar, Niv Buchbinder, TH Hubert Chan, Shahar Chen, Ilan Reuven Cohen, Anupam Gupta, Zhiyi Huang, Ning Kang, Viswanath Nagarajan, Joseph Seffi Naor, and Debmalya Panigrahi. Online algorithms for covering and packing problems with convex objectives. In *Proc. Symposium on Foundations of Computer Science (FOCS)*, 2016.

4 Francis Bach. Submodular functions: from discrete to continuous domains. *Mathematical Programming*, pages 1–41, 2016.

5 Francis Bach et al. Learning with submodular functions: A convex optimization perspective. *Foundations and Trends® in Machine Learning*, 6(2-3):145–373, 2013.

6 Maria-Florina Balcan and Nicholas J. A. Harvey. Learning submodular functions. In *Proc. Machine Learning and Knowledge Discovery in Databases*, pages 846–849, 2012.

7 Eric Balkanski and Yaron Singer. A lower bound for parallel submodular minimization. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, pages 130–139, New York, NY, USA, 2020. Association for Computing Machinery.

8 Etienne Bamas, Andreas Maggiori, and Ola Svensson. The primal-dual method for learning augmented algorithms. In *Proc. 34th Conference on Neural Information Processing Systems*, 2020.

9 Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis.* Cambridge University Press, 2005.

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

11 Niv Buchbinder and Joseph Naor. Online primal-dual algorithms for covering and packing. *Mathematics of Operations Research*, 34(2):270–286, 2009.

12 Ioannis Caragiannis. Better bounds for online load balancing on unrelated machines. In *Proc. 19th ACM-SIAM Symposium on Discrete Algorithms*, pages 972–981, 2008.

13 Robert D. Carr, Lisa K. Fleischer, Vitus J. Leung, and Cynthia A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Proc. 11th Symposium on Discrete Algorithms*, pages 106–115, 2000.

14 Michele Conforti and Gérard Cornuéjols. Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the rado-edmonds theorem. *Discrete applied mathematics*, 7(3):251–274, 1984.

15 Michel X Goemans, Nicholas JA Harvey, Satoru Iwata, and Vahab Mirrokni. Approximating submodular functions everywhere. In *Proc. 20th Symposium on Discrete algorithms*, pages 535–544, 2009.

16 Sreenivas Gollapudi and Debmalya Panigrahi. Online algorithms for rent-or-buy with expert advice. In *International Conference on Machine Learning*, pages 2319–2327, 2019.

17 Elena Grigorescu, Young-San Lin, Sandeep Silwal, Maoyuan Song, and Samson Zhou. Learning-augmented algorithms for online linear and semidefinite programming. In *Advances in Neural Information Processing Systems*, 2022.

18 Lin Gu, Jingjing Cai, Deze Zeng, Yu Zhang, Hai Jin, and Weiqi Dai. Energy efficient task allocation and energy scheduling in green energy powered edge computing. *Future Generation Computer Systems*, 2019.

19 Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. Learning-based frequency estimation algorithms. In *Proc. Conference on Learning Representations*, 2019.

20 Satoru Iwata, Lisa Fleischer, and Satoru Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM (JACM)*, 48(4):761–777, 2001.

21 Rishabh K Iyer, Stefanie Jegelka, and Jeff A Bilmes. Curvature and optimal algorithms for learning and minimizing submodular functions. In *Advances in Neural Information Processing Systems*, pages 2742–2750, 2013.
22 Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *Proc. Conference on Management of Data*, pages 489–504, 2018.
23 Ravi Kumar, Manish Purohit, and Zoya Svitkina. Improving online algorithms via ml predictions. In *Proc. 32nd Conference on Neural Information Processing Systems*, pages 9684–9693, 2018.
24 Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online scheduling via learned weights. In *Proc. Symposium on Discrete Algorithms*, pages 1859–1877, 2020.
25 Thodoris Lykouris and Sergei Vassilvtiskii. Competitive caching with machine learned advice. In *International Conference on Machine Learning*, pages 3296–3305, 2018.
26 Michael Mitzenmacher. A model for learned bloom filters, and optimizing by sandwiching. In *Proc. Conference on Neural Information Processing Systems*, pages 464–473, 2018.
27 Michael Mitzenmacher. Scheduling with predictions and the price of misprediction. In *Proc. 11th Innovations in Theoretical Computer Science Conference*, 2020.
28 Michael Mitzenmacher, Sergei Vassilvitskii, and Tim Roughgarden. *Beyond the Worst-Case Analysis of Algorithms*, chapter Algorithms with Predictions. Cambridge University Press, 2020.
29 Dhruv Rohatgi. Near-optimal bounds for online caching with machine learned advice. In *Proc. Symposium on Discrete Algorithms*, pages 1834–1845, 2020.
30 Tim Roughgarden. Intrinsic robustness of the price of anarchy. *Journal of the ACM*, 62(5):32, 2015.
31 Tim Roughgarden. Beyond worst-case analysis. *Communications of the ACM*, 62(3):88–96, 2019.
32 Tim Roughgarden. *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press, 2020.
33 Maxim Sviridenko, Jan Vondrák, and Justin Ward. Optimal approximation for submodular and supermodular optimization with bounded curvature. *Mathematics of Operations Research*, 2017.
34 Nguyen Kim Thang. Online primal-dual algorithms with configuration linear programs. In *Proc. 31st International Symposium on Algorithms and Computation*, 2020.
35 Jan Vondrák. Submodularity and curvature: The optimal algorithm. *RIMS Kokyuroku Bessatsu*, 2010.
36 David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge University Press, 2011.

Appendix

A Omitted proofs in Section 3

▶ **Lemma 4.** *Let e be an arbitrary resource. At any moment τ during the execution of the algorithm, when t constraints have already been released, it always holds that*

$$x_e \geq \frac{\eta}{b_e^{t_e^*}(A)\cdot d}\left[\exp\left(\frac{\ln(1+2d^2/\eta)}{\beta_e}\cdot\sum_{A:e\notin A}\sum_{t'\leq t} b_e^{t'}(A)\cdot\alpha_A^{t'}\right)-1\right]$$

(where recall that $b_e^{t_e^*}(A)$ is defined in the algorithm on line 16).

Proof. Let us fix a resource e and prove the lemma by induction. At the beginning of the execution, when no constraint has been released yet, both sides of the lemma are 0. Let us assume that the lemma holds until the release of the t^{th} constraint $\sum_e a_e^t x_e \geq 1$. Consider a moment τ during the algorithm's execution and let A^* be the current set of resources e' such that $x_{e'} = 1$. If at time τ, $x_e = 1$ then by the algorithm's design, the set A^* has been updated such that $e \in A^*$. Consequently, the increasing rates of both sides in the lemma inequality are 0. In the remaining part of the proof, let us assume that $x_e < 1$. We recall that by the algorithm's design, $\beta_e \geq \frac{1}{\lambda}\nabla_e F(\boldsymbol{x})$. We consider two cases $\beta_e > \frac{1}{\lambda}\nabla_e F(\boldsymbol{x})$ and $\beta_e = \frac{1}{\lambda}\nabla_e F(\boldsymbol{x})$.

Case 1: $\beta_e > \frac{1}{\lambda}\nabla_e F(\boldsymbol{x})$. In this case, by the algorithm's design, the value of β_e remains unchanged at time τ (line 10) ($\frac{\partial\beta_e}{\partial\tau} = 0$). The lemma's right-hand side's derivative according to τ is

$$\begin{aligned}
&\sum_{t'\leq t}\frac{\partial\alpha_{A^*}^{t'}}{\partial\tau}\cdot\frac{b_e^{t'}(A^*)\ \eta}{b_e^{t_e^*}(A)\ d}\cdot\frac{\ln(1+2d^2/\eta)}{\beta_e}\cdot\exp\left(\frac{\ln(1+2d^2/\eta)}{\beta_e}\cdot\sum_{A:e\notin A}\sum_{t'\leq t} b_e^{t'}(A)\ \alpha_A^{t'}\right)\\
&\leq \frac{\partial\alpha_{A^*}^{t}}{\partial\tau}\cdot\frac{b_e^{t}(A^*)\ \eta}{b_e^{t_e^*}(A)\ d}\cdot\frac{\ln(1+2d^2/\eta)}{\beta_e}\cdot\left(\frac{b_e^{t_e^*}(A)\ d}{\eta}\ x_e+1\right)\\
&= \frac{1}{\lambda\ln(1+2d^2/\eta)}\cdot\frac{b_e^{t}(A^*)\ \eta}{b_e^{t_e^*}(A)\ d}\cdot\frac{\ln(1+2d^2/\eta)}{\beta_e}\cdot\left(\frac{b_e^{t_e^*}(A)\ d}{\eta}\ x_e+1\right)\\
&\leq \frac{b_e^{t}(A^*)\ x_e}{\lambda\ \beta_e}+\frac{\eta}{\lambda\ \beta_e\ d}\\
&\leq \frac{\partial x_e}{\partial\tau}
\end{aligned}$$

In the first inequality, we use the induction hypothesis and $\frac{\partial\alpha_{A^*}^{t}}{\partial\tau} > 0$ and $\frac{\partial\alpha_{A^*}^{t'}}{\partial\tau} \leq 0$ for $t' < t$ and $\frac{\partial\beta_e}{\partial\tau} = 0$. The equality follows the increasing rate of $\alpha_{A^*}^{t}$. The last inequality is due to the increasing rate of x_e. The rate on the left-hand side is always larger than on the right-hand side, so the lemma inequality holds.

Case 2: $\beta_e = \frac{1}{\lambda}\nabla_e F(\boldsymbol{x})$. In this case, by the algorithm's design, $\frac{1}{\lambda}\nabla_e F(\boldsymbol{x})$ is locally non-decreasing at τ (since otherwise, by line 10, β_e is not maintained to be equal to $\frac{1}{\lambda}\nabla_e F(\boldsymbol{x})$). Therefore, $\frac{\partial\beta_e}{\partial\tau} \geq 0$ and so $\partial\big(\frac{1}{\beta_e}\big)/\partial\tau \leq 0$. Hence, the derivative of the right-hand side of the lemma inequality according to τ is upper bounded by

$$\sum_{t'\leq t}\frac{\partial\alpha_{A^*}^{t}}{\partial\tau}\cdot\frac{b_e^{t'}(A^*)\ \eta}{b_e^{t_e^*}(A)\ d}\cdot\frac{\ln(1+2d^2/\eta)}{\beta_e}\cdot\exp\left(\frac{\ln(1+2d^2/\eta)}{\beta_e}\cdot\sum_{A:e\notin A}\sum_{t'\leq t} b_e^{t'}(A)\ \alpha_A^{t'}\right)$$

which is bounded by $\frac{\partial x_e}{\partial \tau}$ by the same argument as the previous case. The lemma follows. ◀

B Applications in Section 3

To apply Theorem 1 on specific problems, we need to determine the local-smoothness parameters for the multilinear extension. [34] provided these parameters for some broad classes of functions, in particular for polynomials with non-negative coefficients. Let $g_\ell : \mathbb{R} \to \mathbb{R}$ for $1 \leq \ell \leq L$ be degree-k polynomials with non-negative coefficients and let $f : \{0,1\}^n \to \mathbb{R}^+$ be the cost function defined as $f(\mathbf{1}_S) = \sum_\ell b_\ell g_\ell\big(\sum_{e \in S} a_e\big)$ where $a_e \geq 0$ for every e and $b_\ell \geq 0$ for every $1 \leq \ell \leq L$. Then the multilinear extension F of f is $(O(k \ln(d/\eta))^{k-1}, \frac{k-1}{k \ln(1+2d^2/\eta)})$-locally smooth. We will use these parameters to derive the guarantees for the following problems.

B.1 Load Balancing

Problem. Load balancing is a classic problem in discrete optimization with wide-ranging applications (for example, resource management in data centres). This problem revolves around assigning jobs that arrive online to m available unrelated machines while minimizing their maximum load. Each arriving job j reveals its machine dependent execution time p_{ij} where $i \in \{1, m\}$ is the machine's index. The load ℓ_i of machine i is the total processing time of the jobs assigned to it. This load balancing problem is a well understood standard online problem and it has a tight competitive ratio of $\Theta(\log m)$ ([9, 12]).

In our online setting with predictions, the jobs not only arrive with their machine dependent execution time p_{ij}, but their machine dependent prediction as well. Formally, $x_{ij} \in \{0,1\}$ indicates whether job j is assigned to machine i, and the oracle provides $pred(x_{ij}) \in \{0,1\}$. We can formulate the online load balancing problem as a non-linear program. The objective is $\min \max_{i=1}^m \ell_i = \min \max_{i=1}^m \big(\sum_j p_{ij} x_{ij}\big)$, and the constraint is $\sum_{i=1}^m x_{ij} = 1$ which guarantees that each job j is assigned to some machine i. Applying our framework for non-linear programs with covering constraints, Proposition 6 follows.

▶ **Proposition 6.** *Algorithm 1 gives a $O(\frac{1}{1-\eta})$-consistent and $O\big((\log m)\log^2 \frac{m}{\eta}\big)$-robust fractional solution for the load balancing problem.*

Proof. It is known that ∞-norm of a m-dim vector can be approximated by the $(\log m)$-norm, in particular for $m \geq 2$,

$$\|(\ell_1, \ell_2, \ldots, \ell_m)\|_\infty \leq \|(\ell_1, \ell_2, \ldots, \ell_m)\|_{\log m} \leq m^{1/m} \|(\ell_1, \ell_2, \ldots, \ell_m)\|_\infty \leq 2\|(\ell_1, \ell_2, \ldots, \ell_m)\|_\infty.$$

Hence, one can instead consider the objective of minimizing the $(\log m)$-norm of the load vectors while losing a constant factor of 2. More precisely, we consider the $(\log m)$-th power of the $(\log m)$-norm as the objective.

$$\min \sum_{i=1}^m \Big(\sum_j p_{ij} x_{ij}\Big)^{\log m} \qquad \text{s.t.} \qquad \sum_{i=1}^m x_{ij} = 1 \ \forall j$$

The objective function is a polynomial of degree $\log m$. So its multilinear extension is $(O(k \ln(d/\eta))^{k-1}, \frac{k-1}{k \ln(1+2d^2/\eta)})$-locally smooth with $k = \log m$ and $d = m$ (the maximal number of positive coefficients in a constraint). Therefore, applying Theorem 1, the robustness (w.r.t the objective as the $(\log m)$-th power of the $(\log m)$-norm) is $O\big((\log m \log \frac{m}{\eta})^{\log m}\big)$. Getting back to the $(\log m)$-norm objective by taking the $(\log m)$-root, the robustness is $O\big((\log m)\log^2 \frac{m}{\eta}\big)$. Hence, Algorithm 1 is $O(\frac{1}{1-\eta})$-consistent and $O\big((\log m)\log^2 \frac{m}{\eta}\big)$-robust. ◀

B.2 Energy Minimization in Scheduling

Problem. Reducing carbon emissions is a global effort in which energy-efficient algorithms play an essential role. For example, [1] and [18] studied energy-efficient algorithms for scheduling.

Given m unrelated machines, we need to assign jobs that arrive online. Each job j has a release date r_j, a deadline d_j, and a vector of machine dependent processing times p_{ij}. Contrary to performance-oriented scheduling, our goal is to design an assignment policy which can minimize the total energy consumption of the execution. To achieve this, we can adjust the machines' speed $s_{ij}(t)$ during the time interval $[t, t+1)$ for the execution of job j. Every machine i has a non-decreasing energy power function $P_i(\cdot)$. Typically, $P_i(z) = z^{k_i}$ for some constant $k_i \geq 1$. The execution's total energy is $\sum_i \sum_t P(\sum_j s_{ij}(t))$.

In the classic online setting, this problem is well understood: there exists an $O(k^k)$-competitive algorithm ([34]) where $k = \max_i\{k_i\}$ and this bound is tight up to a constant factor ([12]). In our extended study with predictions we represent this problem with the following non-linear program. The objective is $\min \sum_i \sum_t P(\sum_j s_{ij}(t))$ and the constraints are:

$$\sum_{i=1}^{m} x_{ij} = 1, \qquad \sum_{t=r_j}^{d_j-1} s_{ij}(t) \geq p_{ij} x_{ij}, \qquad s_{ij}(t) \geq 0 \quad \forall\ i,\ t$$

where $x_{ij} \in \{0,1\}$ indicates whether job j is assigned to machine i and $s_{ij}(t) \geq 0$ denotes the speed of machine i executing job j during the time interval $[t, t+1)$. The first constraint guarantees that job j is assigned to some machine, and the second one ensures that the job j is completed on time (on the machine where the job is assigned). At the arrival of job j, the prediction provides a solution $\mathit{pred}(x_{ij})$ and a speed $\mathit{pred}(s_{ij}(t))$ for $r_j \leq t \leq d_j - 1$. Using our framework, we can deduce the following result.

▶ **Proposition 7.** *Algorithm 1 gives a $O(\frac{1}{1-\eta})$-consistent and $O\big(k^k \log^k \frac{m}{\eta}\big)$-robust fractional solution for the energy minimization problem.*

Proof. The objective function $\sum_i \sum_t P(\sum_j s_{ij}(t))$ is a polynomial of degree $k = \max_i k_i$; so its multilinear extension is $(O(k \ln(m/\eta))^{k-1}, \frac{k-1}{k \ln(1+2m^2/\eta)})$-locally smooth (the maximal number of positive coefficients in a constraint $d = m$). Therefore, applying Theorem 1, Algorithm 1 provides a $O(\frac{1}{1-\eta})$-consistent and $O\big(k^k \ln^k \frac{m}{\eta}\big)$-robust fractional solution. ◀

B.3 Online Submodular Mimimization

Problem. Submodular minimization is a widespread subject in optimization and machine learning ([20, 5, 4, 7]). Let us consider the problem of minimizing an online monotone submodular function subject to covering constraints. A set-function $f : 2^{\mathcal{E}} \to \mathbb{R}+$ is *submodular* if $f(S \cup e) - f(S) \geq f(T \cup e) - f(T)$ for all $S \subset T \subseteq \mathcal{E}$. Let F be the multilinear extension of a monotone submodular function f. Function F admits two useful properties. First, if f is monotone, then so is F. Second, F is concave in the positive direction, meaning that $\nabla F(\boldsymbol{x}) \geq \nabla F(\boldsymbol{y})$ for all $\boldsymbol{x} \leq \boldsymbol{y}$, where $\boldsymbol{x} \leq \boldsymbol{y}$ is defined as $x_e \leq y_e\ \forall e$.

To apply Algorithm 1, we need to determine the local-smoothness parameters. An important concept in studying submodular functions is the *curvature*. Given a submodular function f, the *total curvature* κ_f ([14]) of f is defined as $\kappa_f = 1 - \min_e \frac{f(\mathbf{1}_{\mathcal{E}}) - f(\mathbf{1}_{\mathcal{E}\setminus\{e\}})}{f(\mathbf{1}_{\{e\}})}$. Intuitively, the total curvature measures how far away f is from being *modular*. This concept of curvature is used to determine both upper and lower bounds on the approximation ratios for many submodular and learning problems (see [14, 15, 6, 35, 21, 33]). The following lemma shows a useful property of the total curvature.

▶ **Lemma 8.** *For any set S, it always holds that*

$$f(\mathbf{1}_S) \geq (1-\kappa_f) \sum_{e \in S} f(\mathbf{1}_{\{e\}}).$$

Proof. Let $S = \{e_1, \dots, e_m\}$ be an arbitrary subset of $\mathcal{E}$. Let $S_i = \{e_1, \dots, e_i\}$ for $1 \leq i \leq m$ and $S_0 = \emptyset$. We have

$$\begin{aligned} f(\mathbf{1}_S) &\geq f(\mathbf{1}_{\mathcal{E}}) - f(\mathbf{1}_{\mathcal{E}\setminus S}) = \sum_{i=0}^{m-1} f(\mathbf{1}_{\mathcal{E}\setminus S_i}) - f(\mathbf{1}_{\mathcal{E}\setminus S_{i+1}}) \geq \sum_{i=1}^{m} f(\mathbf{1}_{\mathcal{E}}) - f(\mathbf{1}_{\mathcal{E}\setminus\{e_i\}}) \\ &\geq (1-\kappa_f) \sum_{i=1}^{m} f(\mathbf{1}_{e_i}) \end{aligned}$$

where the first two inequalities are due to the submodularity of f, and the last inequality follows the definition of curvature. ◀

▶ **Proposition 7.** *Algorithm 1 gives a $O(\frac{1}{1-\eta})$-consistent and $O\big(\frac{\log(d/\eta)}{1-\kappa_f}\big)$-robust fractional solution for the submodular minimization under covering constraints.*

Proof. Let F be the multilinear extension of f. It is sufficient to verify that F is $\big(\frac{1}{1-\kappa_f}, 0\big)$-locally smooth. Recall that, by definition of the multilinear extension, $F(\boldsymbol{x}) = \mathbb{E}\big[f(\mathbf{1}_T)\big]$ where T is a random set such that a resource e appears in T with probability x_e. Moreover, as F is linear in x_e, we have

$$\begin{aligned} \nabla_e F(\boldsymbol{x}) &= F(x_1, \dots, x_{e-1}, 1, x_{e+1}, \dots, x_n) - F(x_1, \dots, x_{e-1}, 0, x_{e+1}, \dots, x_n) \\ &= \mathbb{E}\Big[f\big(\mathbf{1}_{R\cup\{e\}}\big) - f\big(\mathbf{1}_R\big)\Big] \end{aligned}$$

where R is a random subset of resources $N \setminus \{e\}$ such that e' is included with probability $x_{e'}$. Therefore, to prove that F is (λ, μ)-locally-smooth, it is equivalent to show that, for any set $S \subset \mathcal{E}$ and for any vectors $\boldsymbol{x}^e \in [0,1]^n$ for $e \in \mathcal{E}$,

$$\sum_{e \in S} \mathbb{E}\Big[f\big(\mathbf{1}_{R^e\cup\{e\}}\big) - f\big(\mathbf{1}_{R^e}\big)\Big] \leq \lambda f\big(\mathbf{1}_S\big) + \mu \mathbb{E}\Big[f\big(\mathbf{1}_R\big)\Big]$$

where R^e is a random subset of resources $N \setminus \{e\}$ such that e' is included with probability $x^e_{e'}$ and R is a random subset of resources $N \setminus \{e\}$ such that e' is included with probability $\max_{e \in S} x^e_{e'}$.

Indeed, the $\big(\frac{1}{1-\kappa_f}, 0\big)$-local smoothness of F holds due to the submodularity and Lemma 8: for any subsets R^e, we have

$$\sum_{e \in S} \big[f\big(\mathbf{1}_{R^e\cup\{e\}}\big) - f\big(\mathbf{1}_{R^e}\big)\big] \leq \sum_{e \in S} \big[f\big(\mathbf{1}_{\{e\}}\big)\big] \leq \frac{1}{1-\kappa_f} \cdot f\big(\mathbf{1}_S\big)$$

Therefore, applying Theorem 1, the proposition follows. ◀

C Additional experiment results

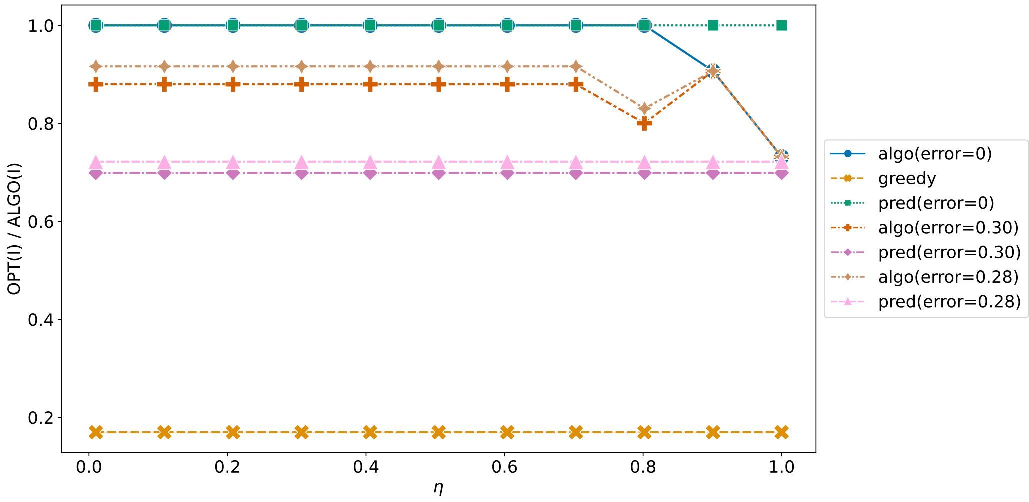

Figure 2 Experiment result. The x-axis show the confidence in the prediction, where 0 means higher confidence. The y-axis show the competitive ratio compared to the optimal offline integral solution. The different colors (also markers) show the result of the algorithm with different prediction error rates and the solutions of the greedy algorithm and the prediction alone. The input graph has 20 vertices, 69 arcs, and 10 requests.

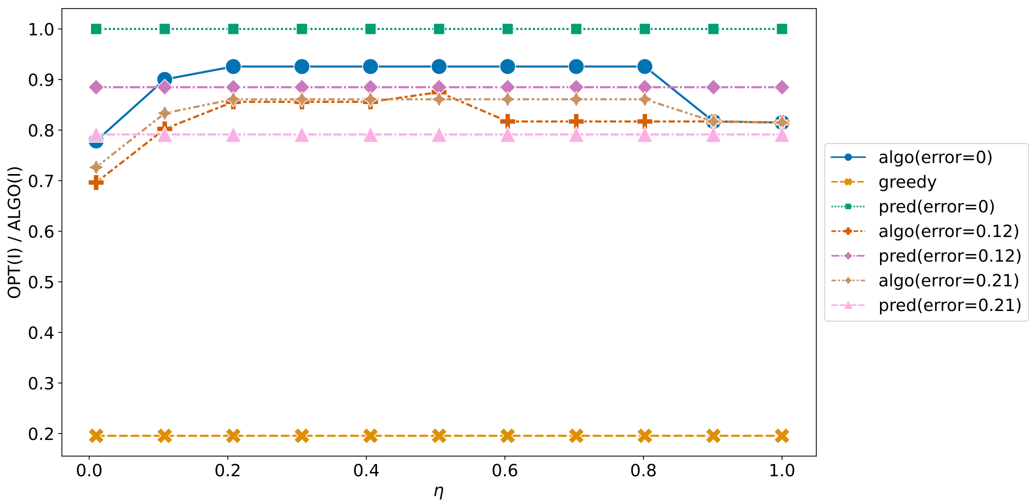

Figure 3 Experiment result. The x-axis show the confidence in the prediction, where 0 means higher confidence. The y-axis show the competitive ratio compared to the optimal offline integral solution. The different colors (also markers) show the result of the algorithm with different prediction error rates and the solutions of the greedy algorithm and the prediction alone. The input graph has 30 vertices, 73 arcs, and 20 requests.