Learning-Augmented Algorithms for Online Linear and Semidefinite Programming

Elena Grigorescu* Young-San Lin† Sandeep Silwal‡
Maoyuan Song§ Samson Zhou▽

October 24, 2022

Abstract

Semidefinite programming (SDP) is a unifying framework that generalizes both linear programming and quadratically-constrained quadratic programming, while also yielding efficient solvers, both in theory and in practice. However, there exist known impossibility results for approximating the optimal solution when constraints for covering SDPs arrive in an online fashion. In this paper, we study online covering linear and semidefinite programs in which the algorithm is augmented with advice from a possibly erroneous predictor. We show that if the predictor is accurate, we can efficiently bypass these impossibility results and achieve a constant-factor approximation to the optimal solution, i.e., consistency. On the other hand, if the predictor is inaccurate, under some technical conditions, we achieve results that match both the classical optimal upper bounds and the tight lower bounds up to constant factors, i.e., robustness.

More broadly, we introduce a framework that extends both (1) the online set cover problem augmented with machine-learning predictors, studied by Bamas, Maggiori, and Svensson (NeurIPS 2020), and (2) the online covering SDP problem, initiated by Elad, Kale, and Naor (ICALP 2016). Specifically, we obtain general online learning-augmented algorithms for covering linear programs with fractional advice and constraints, and initiate the study of learning-augmented algorithms for covering SDP problems.

Our techniques are based on the primal-dual framework of Buchbinder and Naor (Mathematics of Operations Research, 34, 2009) and can be further adjusted to handle constraints where the variables lie in a bounded region, i.e., box constraints.

1 Introduction

In the classical online model, an input is iteratively given to an algorithm that must make irrevocable decisions at each point in time, while satisfying a number of changing constraints and optimizing a fixed predetermined objective. A common metric for evaluating the quality of an online algorithm is the competitive ratio, which is the ratio between the "cost" of the algorithm and the best cost in hindsight, i.e., an optimal offline algorithm given the entire input sequence in advance. In the context of the minimization problems we study in this paper, an online algorithm is c-competitive if its cost is at most a multiplicative c factor more than

*Purdue University. Supported in part by NSF CCF-1910659, NSF CCF-1910411 and NSF CCF-2228814. E-mail: elena-g@purdue.edu

†University of Melbourne. Work done while at Purdue University. Supported in part by NSF CCF-1910659, NSF CCF-1910411 and NSF CCF-2228814. E-mail: nilnamuh@gmail.com

‡Massachusetts Institute of Technology. Supported by an NSF Graduate Research Fellowship under Grant No. 1745302, NSF TRIPODS program (award DMS-2022448), and Simons Investigator Award. E-mail: silwal@mit.edu

§Purdue University. Supported in part by NSF CCF-1910659, NSF CCF-1910411, NSF CCF-2127806, NSF CCF-2228814 and a Ross-Lynn Award. E-mail: song683@purdue.edu

▽UC Berkeley and Rice University. Work supported by a Simons Investigator Award and by NSF CCF-1815840, and done in part while at Carnegie Mellon University, USA. E-mail: samsonzhou@gmail.com

the cost of the optimal solution. Due to the irrevocable decisions, the changing constraints, or the number of possible different worst-case inputs, many online algorithms have undesirable competitive ratios that are impossible to improve upon without additional assumptions, e.g., [AAA+09].

Due to advances in the predictive ability of machine learning models, a natural approach to overcome these computational barriers is to incorporate models with predictions, e.g. models that predict outcomes based on historical data. Unfortunately, due to the lack of provable guarantees on worst-case input, these predictions can be embarrassingly inaccurate when attempting to generalize to unfamiliar inputs, as shown in [SZS+14], or simply not even satisfy the given constraints [BMS20]. Thus, rather than blindly following an erroneous machine learning predictor, recent focus has shifted to studying algorithms that use the output of these models as *advice*, and guarantee good competitive ratios both when the predictions are accurate, i.e., consistency, and when the predictions are poor, i.e., robustness.

Recently, [BMS20] studied the learning-augmented online set cover problem and related problems, using their linear programming (LP) formulation to incorporate additional advice through a primal-dual approach. One drawback of their seminal work however, is that they assume both *integral* constraints, as well as integral advice, which restricts the modeling capabilities of the framework; it is natural to ask how an online algorithm can be improved when the advice is given in terms of probability distribution or some other meaningful *fractional* values. For example, for the online set cover problem, fractional advice can indicate how likely a set should be chosen instead of the binary decision of whether a set should be chosen or not; for the ski rental problem, the advice can be presented as a probability distribution over the total number of vacation days; in online network connectivity problems, the advice can indicate how likely an edge should be chosen.

In addition, linear programs cannot handle quadratic constraints and thus often fail to capture important aspects of fundamental optimization problems, which motivates the study of more general programs, such as semidefinite programming (SDP). SDP is a unifying framework that generalizes both linear programs and quadratically constrained quadratic programming (QCQP), while also yielding very efficient solvers, both in theory and in practice [VB96].

Preliminaries. For a learning-augmented problem, we are given a confidence parameter $\lambda \in [0, 1]$, where lower values of λ denote higher confidence in the advice, and higher values denote lower confidence. An advice is a suggested solution for the online problem that is given. In the content of optimization problems including linear and semidefinite programming, a solution is a vector consisting of real numbers. We denote APX as the objective value of the online solution obtained by an online algorithm, and compare it with (1) the objective value of the advice, denoted as ADV, and (2) the objective value of the offline optimal solution, denoted as OPT. The consistency and robustness of an online algorithm or solution for a minimization optimization problem are defined as follows.

Definition 1.1. An online solution with objective value APX is $C(\lambda)$-consistent if $\mathsf{APX} \leq C(\lambda)\mathsf{ADV}$. An online algorithm is $C(\lambda)$-consistent if it generates a $C(\lambda)$-consistent solution. Here, $C : [0, 1] \to \mathbb{R}_{\geq 1}$.

Definition 1.2. An online solution with objective value APX is $R(\lambda)$-robust if $\mathsf{APX} \leq R(\lambda)\mathsf{OPT}$. An online algorithm is $R(\lambda)$-robust if it generates a $R(\lambda)$-robust solution. Here, $R : [0, 1] \to \mathbb{R}_{\geq 1}$.

Learning-augmented algorithms for minimization problems consider the advice while approximately minimizing the objective value when the input arrives online. Intuitively, if an advice is accurate and we trust the advice, then we would like the solution to be close to the optimal, so ideally $C(\lambda)$ should approach 1 as λ approaches 0. On the other hand, having λ being close to 1 denotes no trust in the advice, so $R(1)$ should be close to the optimal competitive ratio of the best pure online algorithm.

1.1 Our Contributions

We give a general paradigm for designing learning-augmented algorithms for online covering linear programming [BN09b], which generalizes the set cover problem [BMS20], as well as online covering semidefinite programming [EKN16], with possibly non-integral constraints and advice. Specifically, we present *primal-dual learning-augmented (PDLA)* algorithms for these problems, whose performance is close to the optimal

offline solution when the advice is accurate, and also whose performance is asymptotically close to the optimal oblivious algorithm, if the advice is inaccurate.

Our PDLA algorithms consider the advice while approximately minimizing the objective value when the input arrives online. Our unifying paradigm applies to both online covering linear programs (LP) and online covering semidefinite programs (SDP) described below.

Online covering linear programs. A *covering* LP is defined as follows:

$$\text{minimize } c^T x \text{ over } x \in \mathbb{R}^n_{\geq 0} \text{ subject to } Ax \geq \mathbf{1}. \tag{1}$$

Here, $A \in \mathbb{R}^{m \times n}_{\geq 0}$ consists of m *covering* constraints, $\mathbf{1}$ is a vector of all ones, and $c \in \mathbb{R}^n_{>0}$ denotes the positive coefficients of the linear cost function. We use a_{ij} to denote the i-th row j-th column entry of A and c_j to denote the j-th coordinate of c.

In the online covering problem, the cost vector c is given offline, and each of these covering constraints (rows) is presented one by one in an online fashion, that is, m can be unknown. The goal is to update x in a non-decreasing manner such that all the covering constraints are satisfied and the objective value $c^T x$ is approximately minimized.

A classical example captured by the covering LP (1) is the set cover problem. In this problem, we have a universe $[m]$ and n sets $S_1, S_2, ..., S_n$ which are subsets of $[m]$. Each set S_j where $j \in [n]$ is associated with a cost c_j. The goal is to find a subset of set indices $C \subseteq [n]$, such that (1) the universe is covered by the union of the sets whose corresponding indices are in C, i.e., $\cup_{j \in C} S_j = [m]$ and (2) the cost $\sum_{j \in C} c_j$ is minimized. An LP relaxation can be formulated by having x encoding the indicator vector for the set selection, the columns representing the sets, and the rows representing the universe. In the constraint matrix A, $a_{ij} = 1$ if element i is contained in set S_j, otherwise $a_{ij} = 0$.

The online set cover problem can be naturally captured by the online covering LP (1). We have the sets given offline and the elements arriving online. Upon the arrival of an element, the information that which sets contain the arriving element is revealed (as a row of A), and we have to irrevocably pick the sets to cover the universe. The $O(\log m \log n)$-competitive online algorithm introduced in [AAA+09, BN09a] solves the online set cover problem by incrementing the indicator vector x in the covering LP and rounding the fractional solution online by a threshold-based approach. The $O(\log m)$ factor is from the integrality gap of the covering LP while the $O(\log n)$ factor is from the competitiveness of the online algorithm.

An $O(\log n)$-competitive algorithm for online covering LPs was presented in [BN09b], which simultaneously solves both the primal covering LP (1) and the dual *packing* LP (2), defined as follows:

$$\text{maximize } \mathbf{1}^T y \text{ over } y \in \mathbb{R}^m_{\geq 0} \text{ subject to } A^T y \leq c. \tag{2}$$

The analysis in [BN09b] crucially uses LP-duality and strong connections between the two solutions to argue that they are both nearly optimal. The covering solution x is an exponential function of the packing solution y and both x and y are monotonically increasing. The problem naturally extends to the setting that relies on a *separation oracle* to retrieve an unsatisfied covering constraint where the number of constraints can be unbounded. However, as the framework in [BN09b] fixes all violating constraints, each arriving constraint might be slightly violated so that each individual fix may require a diminishingly small adjustment. Consequently the algorithm may have to address exponentially many constraints. The framework was later modified in [GLQ21] which guarantees that addressing polynomially many constraints suffices.

In the learning-augmented problem, we are given a confidence parameter $\lambda \in [0, 1]$ and $x' \in \mathbb{R}^n_{\geq 0}$ served as a fractional advice for LP (1). However, we do not have the guarantees about the advice x'. More specifically, the objective value of the advice $c^T x'$ could be a horrendous approximation to the optimal objective value of LP (1) or x' might not even satisfy the constraints.

We first show an efficient, consistent, and robust PDLA algorithm for the online covering LP (1). We use the condition number κ to denote the upper bound for the ratio between the maximum positive entry and the minimum positive entry for each fixed column of A. For ease of presentation, we assume that x' is *feasible*, i.e., there are no violating constraints caused by the advice x'.

Theorem 1.3 (Informal). *Given a feasible advice $x' \in \mathbb{R}^n_{\geq 0}$ for LP (1) with confidence parameter $\lambda \in [0,1]$, there exists an $O\left(\frac{1}{1-\lambda}\right)$-consistent and $O\left(\log \frac{\kappa n}{\lambda}\right)$-robust online algorithm for the online covering LP problem that encounters polynomially many violating constraints.*

The formal version of Theorem 1.3 (Theorem 2.1), which addresses the case when x' is infeasible for LP (1), is presented in Section 2. We remark that Theorem 1.3 implies that when $\kappa = \text{poly}(n)$, the algorithm is $\log(n/\lambda)$-robust.

Online covering semidefinite programs. We generalize our approach for learning-augmented covering LPs to handle a more expressive family of optimization problems, namely, covering semidefinite programs.

First, we introduce some standard notation. A matrix $A \in \mathbb{R}^{d\times d}$ is said to be *positive semidefinite* (PSD), i.e., $A \succeq 0$, if $v^T A v \geq 0$ for every vector $v \in \mathbb{R}^d$, or equivalently, all the eigenvalues of A are non-negative. If A is PSD and symmetric, then it is called *symmetric positive semidefinite (SPSD)*. A partial order over SPSD matrices in $\mathbb{R}^{d\times d}$ can be induced such that $A \succeq B$ if and only if $A - B \succeq 0$.

The setting of a covering SDP problem is as follows.

$$\text{minimize } c^T x \text{ over } x \in \mathbb{R}^n_{\geq 0} \text{ subject to } \sum_{j=1}^{n} A_j x_j \succeq B \tag{3}$$

where $A_1, \ldots, A_n \in \mathbb{R}^{d\times d}$ and $B \in \mathbb{R}^{d\times d}$ are SPSD matrices and $c \in \mathbb{R}^n_{>0}$.

In the online covering SDP problem introduced in [EKN16], we have the matrices $A_1, ..., A_n$ and the cost vector c given offline. In each *round* $i \in [m]$ where m can be unknown, we are given a new SPSD matrix $B^{(i)}$ satisfying $B^{(i)} \succeq B^{(i-1)}$. The goal is to cover $B^{(i)}$ using a linear combination $x_1, \ldots, x_n$ of the matrices $A_1, \ldots, A_n$, so that $\sum_{j=1}^n x_j A_j \succeq B^{(i)}$, while minimizing the cost $c^T x$. Moreover, we must update x in a non-decreasing manner, so that once some amount of the matrix A_j is used in the covering at a round i, then it must be used in all subsequent coverings in later rounds. The online covering SDP problem and its dual in round i are as follows:

$$\text{minimize } c^T x \text{ over } x \in \mathbb{R}^n_{\geq 0} \text{ subject to } \sum_{j=1}^{n} A_j x_j \succeq B^{(i)} \tag{4}$$

$$\text{maximize } B^{(i)} \otimes Y \text{ over } Y \succeq 0 \text{ subject to } A_j \otimes Y \leq c_j \forall j \in [n] \tag{5}$$

where we use $A \otimes B$ to denote the Frobenius product of A and B, i.e., $A \otimes B = \sum_{i,j} A_{i,j} B_{i,j} = \text{trace}(A^T B)$.

We remark that the formulation of online covering SDP (4) generalizes online covering LP (1) when the constraint matrix is known offline but there is no guarantee which covering constraint (row) will arrive. In particular, the SDP formulation for online set cover with n sets and d elements all given offline (but without the knowledge of which elements arrive and their order) is the following: we define matrices $A_1, \ldots, A_n \in \{0,1\}^{d\times d}$ where A_j is a diagonal matrix whose diagonal is simply the indicator vector for the j-th set across the d elements, i.e., entry (k,k) of A_j is 1 if and only if set j contains element k. The matrices $B^{(i)}$ encode the variables that must be covered in round i, so that $B^{(0)}$ is the all zeros matrix and $B^{(i)} - B^{(i-1)}$ is the all-zeros matrix except with a single one in entry (k,k) for the variable k that must be newly covered in round i. No online SDP algorithm can achieve competitive ratio $o(\log n)$ because even if fractional sets are allowed, no online algorithm can achieve competitive ratio better than $O(\log n)$ for the online set cover problem [BN09a].

An optimal $O(\log n)$-competitive online algorithm was presented in [EKN16]. Similar to online covering LPs, an important idea in this line of work is to use weak duality and the strong connections between the primal and the dual solutions. Observe that if x and Y are feasible solutions for the primal and the dual, then

$$c^T x \geq \sum_{j=1}^{n} (A_j \otimes Y) x_j = \left(\sum_{j=1}^{n} (A_j x_j)\right) \otimes Y \geq B^{(i)} \otimes Y, \tag{6}$$

and hence the primal and the dual satisfy weak duality.

In the learning-augmented problem, we are given a confidence parameter $\lambda \in [0,1]$ and a vector $x' \in \mathbb{R}^n_{\geq 0}$ that serves as advice for the linear combination $x_1, \ldots, x_n$ that the algorithm should use for the optimal solution. We have no guarantees about the advice. More specifically, the objective value of the advice $c^T x'$ could be a terrible approximation to the optimal objective value of SDP (4) or x' might not even be feasible.

We use κ to denote the ratio of the largest positive eigenvalue to the smallest positive eigenvalue of the matrices $A_1, \ldots, A_n, B^{(1)}, \ldots, B^{(m)}$ and achieve the following.

Theorem 1.4 (Informal). *Given a feasible advice $x' \in \mathbb{R}^n_{\geq 0}$ for SDP (4) with confidence parameter $\lambda \in [0,1]$, there exists a polynomial time, $O\left(\frac{1}{1-\lambda}\right)$-consistent, and $O\left(\log \frac{\kappa n}{\lambda}\right)$-robust online algorithm for the online covering SDP problem.*

The formal version of Theorem 1.4 (namely, Theorem 3.1), which addresses the case when x' is infeasible for SDP (4), is presented in Section 3. We remark that Theorem 1.4 implies that we can achieve a constant factor approximation to the optimal solution when the advice is accurate ($O(1)$-competitive), which breaks the known $\Omega(\log n)$ competitive ratio obtained by the oblivious online algorithm for covering SDP in [EKN16]. Moreover, for $\kappa = \text{poly}(n)$, we match the optimal approximation ratio of $O(\log n)$ up to constants when the advice is arbitrarily bad.

Adding box constraints. In both the LP and SDP case, it is natural to have the requirement that the variables must lie in a bounded region. Defining the bounded region can be done using "box constraints". The setting for the covering LP with box constraints is in the following form.

$$\text{minimize } c^T x \text{ over } x \in [0,1]^n \text{ subject to } Ax \geq \mathbf{1}. \tag{7}$$

We note that this problem might not have any feasible solution. The upper bound is set to one without loss of generality. Suppose $x_j \in [0, u_j]$, then we can scale c_j and the entries in column j by dividing u_j. Again, in the online problem, the covering constraints arrive one at a time. The goal is to update x in a non-decreasing manner subject to each coordinate of x being capped at 1, and approximately minimize the objective $c^T x$. An $O(\log n)$-competitive algorithm for online covering with box constraints was obtained in [BN09b].

For the learning-augmented variant, we assume that the advice $x' \in [0,1]^n$. Our bound is in terms of a notion of *sparsity* s of matrix A, which we define formally in Theorem 2.4.

Theorem 1.5 (Informal). *Given a feasible advice $x' \in [0,1]^n$ for LP (7) with confidence parameter $\lambda \in [0,1]$, there exists an $O\left(\frac{1}{1-\lambda}\right)$-consistent and $O\left(\log \frac{s}{\lambda}\right)$-robust online algorithm for the online covering LP problem with box constraints that encounters polynomially many violating constraints.*

The formal version of Theorem 1.5 (namely, Theorem 2.4), which addresses the case when x' is infeasible for LP (7), is presented in Section 2.1. This result recovers the bound of the learning-augmented algorithm for the more restricted online set cover problem in [BMS20], where $A \in \{0,1\}^{m \times n}$, and the value of s there is the same as row sparsity (i.e., the maximum number of non-zero entries of any row). We emphasize that our algorithm also considers *fractional* advice, even for the online set cover problem.

Similarly, the setting for online covering SDP with box constraints is in the following form.

$$\text{minimize } c^T x \text{ over } x \in [0,1]^n \text{ subject to } \sum_{j=1}^{n} A_j x_j \succeq B^{(i)}. \tag{8}$$

Again, we assume without loss of generality that the upper bound of x_j is one and our goal is to update x in a non-decreasing manner such that $c^T x$ is approximately minimized. An $O(s)$-competitive algorithm for online covering SDP with box constraints was presented in [EKN16] where s is the sparsity of the SDP problem depending on the A_j's and $B^{(i)}$'s. The sparsity notion coincides with the maximum row sparsity when the SDP is used for capturing covering LPs. We use the same sparsity notion as [EKN16] and show the following theorem for the learning-augmented problem when the advice $x' \in [0,1]^n$ is given.

Theorem 1.6 (Informal). *Given a feasible advice $x' \in [0,1]^n$ for SDP (8) with confidence parameter $\lambda \in [0,1]$, there exists a polynomial time, $O\left(\frac{1}{1-\lambda}\right)$-consistent, and $O\left(\log \frac{s}{\lambda}\right)$-robust online algorithm for the online covering SDP problem with box constraints.*

The formal version of Theorem 1.6 (namely, Theorem 3.2), which addresses the case when x' is infeasible for SDP (8), is presented in Section 3.1. In Table 1, we summarize the state-of-the-art by comparing the most related results from the literature to our framework. We further refer the reader to the high-level technical approach in Section 1.2.

Paper	Problem	Approximation Guarantee	Approach
[BN09a]	online covering LP	with and without box constraints: $O(\log n)$-competitive	continuous guess-and-double
[EKN16]	online covering SDP	without box constraints: $O(\log n)$-competitive with box constraints: $O(\log s)$-competitive	continuous guess-and-double efficient updating
[BMS20]	learning-augmented online set cover	without box constraints: $O(1/(1-\lambda))$-consistent $O(\log(d/\lambda))$-robust	discretized
[GLQ21]	online covering LP	without box constraints: $O(\log n)$-competitive	continuous guess-and-double efficient updating
This Work	learning-augmented online covering LP and SDP with fractional advice	without box constraints: $O(1/(1-\lambda))$-consistent $O(\log(\kappa n/\lambda))$-robust with box constraints: $O(1/(1-\lambda))$-consistent $O(\log(s/\lambda))$-robust	continuous guess-and-double efficient updating

Table 1: Summary of the competitive, consistency, and robustness ratios. We assume that the advice is feasible for the learning-augmented problems. Here, n refers to the number of sets or variables, $\lambda \in [0,1]$ refers to the confidence parameter, κ refers to the condition number, d refers to row sparsity, and s refers to sparsity. We note that online covering LP with box constraints generalizes online set cover with $s = d$. In [EKN16], the guess-and-double scheme is not used for online SDP covering with box constraints.

Applications. We emphasize that our framework uses a continuous approach that is amenable to other learning-augmented optimization problems, and supports fractional advice, which may be interpreted as probabilities. For example, as in [EKN16, AW02, WX06], our framework for covering SDPs may be applied to the quantum hypergraph covering problem. We apply our PDLA algorithm for covering LPs with box constraints in order to obtain online algorithms for: (1) the fractional online set cover problem with fractional advice, and for (2) the online group Steiner tree problem on trees, where a min-cut algorithm is used as a separation oracle to retrieve violating constraints. Our learning-augmented solver for the group Steiner problem on trees can be employed as a black-box for other related problems, including group Steiner tree on general graphs, multicast problem on trees, and the non-metric facility location problem [AAA+06].

1.2 Overview of Our Techniques

We now give a technical overview of our algorithms and describe how both our algorithms for covering LPs and SDPs are guided by several common underlying principles.

For each update, while there exists a violating constraint:

1. Determine a violating constraint.
2. Acquire a "growth rate" for each variable depending on its coefficient in the violating constraint, the corresponding cost, and the advice.
3. Use a guess-and-double approach to determine how fast each of the variables are increased by their growth rates.
4. Increase the variables continuously until the constraint is satisfied.

Figure 1: Summary of our framework

Previous approaches. A natural starting point would be the PDLA algorithm for online set cover by [BMS20], who adopted the primal-dual approach in [BN09a] to incorporate external advice. We recall that in the covering LP formulation of the online set cover problem, each row denotes an element and each column denotes a set. The constraint matrix has entries that are either zero or one. An entry is one if and only if the element (row) belongs to the set (column). Additionally, for the online set cover problem considered in [BMS20], each set is either included in the advice or not, i.e., each coordinate of the suggested indicator vector for the set selection is either one or zero. While it seems plausible that one could extend the *discretized* approach of [BMS20] to handle general coefficients in the constraint matrix, i.e., the online covering LP problem, it is unclear how the growth rates of the variables can be adjusted to guarantee dual feasibility. This is because the positive coefficients in every covering constraint (all with value one) are *balanced* in the online set cover problem, which turns out to be a crucial ingredient to argue dual feasibility by the discretized approach, but we do not have this guarantee for general covering LPs with arbitrary positive coefficients. Instead, we use a different framework inspired by the classical online algorithm literature, e.g., [BN09b, EKN16]. We present a short summary of our framework in Figure 1 and describe it in more details below.

Continuous updates. Each time a new constraint arrives, we *continuously* increase the variables until the constraint is satisfied. We adjust this growth rate of each variable based on its cost in the objective linear function, its coefficient in the arriving constraint, and the advice: a variable is increased at a slower rate if its cost is more expensive, its coefficient in the constraint has a smaller value, or it is less recommended by the predictor. The introduction of fractional values in the advice is the main technical obstacle of our setting. In particular, our algorithm must behave differently in the case where a primal variable has not reached the fractional value recommended by the advice compared to the case where it has reached the recommended value, but the solution does not satisfy all constraints. By contrast, in the integral advice setting of [BMS20], the recommendation value always coincides with the limit at one. To this end, once the variable reaches the recommended value, our algorithms judiciously decelerate the growth of the variable.

Guess-and-double. However, by allowing the coefficients of the constraint matrix to be arbitrary, the optimal objective value OPT can be arbitrary and we need a nice estimate for this. Thus, we adopt the *guess-and-double* technique, e.g., [BN09b, EKN16, GLQ21], where the algorithm is executed in *phases*, so that in each phase we propose a lower-bound estimate of OPT, and the algorithm enters the next phase when the value exceeds our estimate. Note that such techniques are not necessary for [BMS20], as their assumption of coefficients in $\{0, 1\}$ implicitly provided bounds on OPT.

Efficient updating. In more general applications, each arriving update may induce a large or even infinite number of constraints, such as an infinite number of directions induced by an SDP constraint. But now if we sequentially choose a violating constraint and satisfy the constraint exactly as in [BMS20], then there is

no guarantee that we will satisfy all the constraints in a small number of iterations. Thus another technique we adopt to ensure efficiency in conjunction with the guess-and-double technique is to satisfy each arriving constraint by a factor of 2. That is, we instead continue to increment the primal variables until the violating constraint is satisfied by a factor of 2, which ensures that at least one primal variable is doubled, which also implies an upper-bound on the number of violating constraints that must be considered.

Showing robustness and consistency. With the introduction of general coefficients within many components of our LP formulation, the robustness analysis in [BMS20] is no longer applicable, so instead we adapt the primal-dual analysis in [BN09b] for general covering LP problems. In particular, we deal with the general coefficients via a delicate telescoping argument for dual feasibility, since we tune and change the growth rates multiple times even within the same phase.

Towards obtaining the consistency bound, we partition the growth rate based on whether the variable has exceeded the value in the advice, and argue that the growth rate not credited to the advice is at most a certain factor of the growth rate credited to the advice, similar to the line of the argument presented in [BMS20].

Extending to online covering SDPs with advice. In the SDP case, we have arriving matrices rather than arriving elements so that at each time, we need to cover a new PSD matrix $B^{(i)}$ that can be *larger* than the previous PSD matrix $B^{(i-1)}$ in an infinite number of directions. We repeatedly look at the direction with the largest mass that needs to be covered, i.e., the largest eigenvector v of $B^{(i)} - \sum_{j=1}^{n} A_j x_j$. Then to cover the direction v, we set the growth rate of the coefficient of each matrix A_j proportional to the amount that the matrix *aligns* with v, i.e., proportional to $v^T A_j v = A_j \otimes V$, where $V = vv^T$ and $\otimes$ is the Frobenius product. Unfortunately, it does not suffice to cover v alone – there may be many other directions for which $B^{(i)} - \sum_{j=1}^{n} A_j x_j$ is not covered. However, as we satisfy the *implicit linear violating constraint* by a factor of 2, the amount of vectors we have to cover is similarly upper-bounded as in the aforementioned approach for the online covering LP problem.

Lastly, we remark that our unifying framework can be naturally adopted to any online problem that has a covering LP or SDP formulation, equipped with a fractional advice and a confidence parameter.

1.3 Additional Background and Related Work

Learning-augmented algorithms. There has been extensive work in incorporating machine-learned predictions in algorithmic design. Machine-learned predictions to enhance the performance of online algorithms were studied in learning-augmented set cover [BMS20], ski rental [PSK18], and caching [LV18, AGKP22]. [LV18, Roh20] showed that an accurate predictor could be leveraged to provide competitive ratios better than the limits of classical online algorithms for online caching, while subsequent learning-augmented algorithms studied scheduling [LLMV20], ski rental [PSK18, GP19], nearest neighbor search [DIRW20], clustering [EFS+22], triangle counting [CEI+22], frequency estimation [HIKV19], and other algorithmic and data structural problems [Mit18, BMS20, BMRS20, WZ20, JLL+20, DKT+21, EIN+21, ACE+20, AGKK20, AGKP22, ND21].

Another direction on this line of research is the stochastic setting, where the input is drawn from a known distribution. This includes online stochastic matching [FMMM09], online graph optimization [APT22, KDZ+17], and other online problems [Mit18, MNS12]. These models differ from ours since we solely consider a given fractional advice (that might have a distribution interpretation) as a solution of the optimization problem instead of making assumptions on the input distribution.

More recently, [AGKP22] presented a model for solving online covering LPs with multiple predictions, but their model assumes that, unlike the model used in this work and [BMS20], the prediction(s) is not given upfront, and instead upon the arrival of each constraint, a feasible way of satisfying that constraint is presented. Additionally, their analysis compares the solution presented by the learning-augmented algorithm

with benchmark solutions that are consistent with the predictions, and guarantees robustness independently, in contrast to our definition of a confidence parameter to parameterize both confidence and robustness.

[ND21] followed the framework presented in [BMS20] and presented a solution to online packing LPs, in complement to our contributions on online covering LPs. Their methods closely resemble that of [BMS20] and this paper, but requires the advice to be integral, while we extend the framework and generalize to allow for fractional advice. Their work also generalizes online packing LPs to allow non-linear objective functions, which leaves room for potentially more optimal algorithms tailored to linear objectives, avoiding loss in generality.

Classical online algorithms with covering LP formulations. One of the most classical online problems is the online set cover problem, solved in the seminal work of [AAA+09] by implicitly using the primal-dual technique of Goemans [GW95]. The approach was extended to network optimization problems in undirected graphs in [AAA+06], ski rental [KMMO94], and paging [BBN12], then abstracted and generalized to a broad LP-based primal-dual framework for online covering and packing LPs in [BN09b]. We refer the reader to the excellent survey by Buchbinder and Naor [BN09a].

Other variants of online covering and packing LP problems. The main focus of our framework is on solving fundamental optimization problems in the online setting with advice. There are other variants of online covering and packing LP problems without advice, including optimizing convex objectives [ABC+16], optimizing ℓ_q-norm objectives [SN20], mixed covering and packing LPs [ABFP13], and sparse integer programs [GN14]. All these frameworks employ the powerful primal-dual technique to ensure the competitiveness of the online algorithm.

Alternative learning augmented algorithms in the online model. Subsequent to our work, a significantly simpler algorithm with tighter qualitative guarantees was brought to our attention by Roie Levin. For completeness, we describe the algorithm in Appendix B, but we emphasize that the algorithm is due to Roie Levin and is included here with his permission. Nevertheless, we expect that the techniques and analysis that we introduce in this paper may be of independent interest for other related problems or settings, such as the advice being adaptive, or in settings of multiple experts. We believe that understanding the full power of the techniques developed in this paper is an intriguing direction for further research in the still emerging area of learning-augmented algorithms.

1.4 Organization

In Section 2, we present the PDLA algorithms for online covering LPs, prove Theorems 1.3 and 1.5, and show the applications on fractional online set cover with fractional advice and group Steiner tree on trees. In Section 3, we present the PDLA algorithms for online covering SDP and prove Theorems 1.4 and 1.6. We show our experimental evaluations in Section 4. In Appendix A, we show the learnability of the covering SDP problem when the input is drawn from a particular distribution, which might be of independent interest.

2 PDLA Algorithms for Online Covering Linear Programs

In this section, we prove Theorems 1.3 and 1.5. Namely, we present efficient, consistent, and robust PDLA algorithms for online covering LPs. We recall that the covering LP (1) is the following.

$$\text{minimize } c^T x \text{ over } x \in \mathbb{R}^n_{\geq 0} \text{ subject to } Ax \geq \mathbf{1}.$$

Here, $A \in \mathbb{R}^{m \times n}_{\geq 0}$ consists of m covering constraints, $\mathbf{1}$ is a vector of all ones, and $c \in \mathbb{R}^n_{>0}$ denotes the positive coefficients of the linear cost function.

In the online problem, the covering constraints (rows) are presented one at a time, so m can be unknown. The cost c is given offline. We say that a new *round* starts when a new covering constraint arrives. The goal

is to update x in a non-decreasing manner such that x is feasible and the objective $c^T x$ is approximately minimized. In the learning-augmented problem, we are also given an advice $x' \in \mathbb{R}^n_{\geq 0}$. The goal is to further consider the advice x' to obtain a consistent (compared to the advice) and robust (compared to the optimal) solution x.

Recall that an important idea is to simultaneously consider the dual *packing* LP (2):

$$\text{maximize } \mathbf{1}^T y \text{ over } y \in \mathbb{R}^m_{\geq 0} \text{ subject to } A^T y \leq c$$

where A^T consists of n packing constraints with an upper bound c given offline.

We use a guess-and-double approach. Let a_{ij} denote the i-th row j-th column entry of A. The algorithm works in *phases*. We estimate a lower bound $\alpha(r)$ for OPT in phase r. In phase 1, let $\alpha(1) \leftarrow \min_{j\in[n]}\{c_j/a_{1j}\}$ be a proper lower bound for OPT. Once the online objective exceeds $\alpha(r)$, we start the new phase $r+1$ from the current violating constraint (let us call it constraint i_{r+1}, in particular, $i_1 = 1$ since in the first phase the first constraint arrives first), and double the estimated lower bound, i.e. $\alpha(r+1) \leftarrow 2\alpha(r)$.

In the beginning of phase r, $x_j^{(r)} \leftarrow \min\{x'_j, \alpha(r)/(2nc_j)\}$. If $x'_j \leq \alpha(r)/(2nc_j)$, then it is possible that $\alpha(r)/(2nc_j)$ is large, so we have to set $x_j^{(r)} = x'_j$ to ensure consistency. On the other hand, if $x'_j \geq \alpha(r)/(2nc_j)$, then it is possible that x'_j is large and the advice is bad, so we have to set $x_j^{(r)} = \alpha(r)/(2nc_j)$ to ensure robustness.[1]

As x must be updated in a non-decreasing manner, the algorithm maintains $\{x_j^{(\ell)}\}_{\ell\in[r]}$, which denotes the value of each variable x_j from phase 1 to phase r, and the value of each variable x_j is actually set to $\max_{\ell\in[r]}\{x_j^{(\ell)}\}$.

Let A_i denote the i-th row of A. In phase r, upon the arrival of constraint i, if the advice adequately covers constraint i, i.e., $A_i x' \geq 1$, then we increase the variables $x_j^{(r)}$ with growth rate

$$\frac{a_{ij}}{c_j}\left(x_j^{(r)} + \frac{\lambda}{A_i\mathbf{1}} + \frac{(1-\lambda)x'_j\mathbf{1}_{x_j^{(r)}<x'_j}}{A_i x'_c}\right)$$

where x'_c is the advice restricted to entries in which the corresponding variable has not reached the advice yet, or equivalently, $x_j^{(r)} < x'_j$. More formally, the j-th coordinate of x'_c is equal to the j-th coordinate of x' if $x_j^{(r)} < x'_j$ and 0 otherwise. $\mathbf{1}_{x_j^{(r)}<x'_j}$ is an indicator variable with value 1 if $x_j^{(r)} < x'_j$ and 0 otherwise. Intuitively, the additive term $\lambda/(A_i\mathbf{1})$ is the contribution credited to the online algorithm while the additive term $(1-\lambda)x'_j\mathbf{1}_{x_j^{(r)}<x'_j}/(A_i x'_c)$ is the contribution credited to the advice x'. We note that

$$\sum_{j=1}^{n}\frac{a_{ij}}{A_i\mathbf{1}} = 1 \text{ and } \sum_{j=1}^{n}\frac{a_{ij}x'_j\mathbf{1}_{x_j^{(r)}<x'_j}}{A_i x'_c} = 1 \tag{9}$$

so the contributions credited to the online algorithm and the advice are normalized and scaled by a factor of λ and $(1-\lambda)$, respectively.

Alternatively if the advice does not cover the constraint enough, i.e., $A_i x' < 1$, we increase $x_j^{(r)}$ with growth rate

$$\frac{a_{ij}}{c_j}\left(x_j^{(r)} + \frac{1}{A_i\mathbf{1}}\right).$$

Namely, since the advice x' is not feasible, we only consider the contribution from the online algorithm.

In order to implement with the proper growth rate, the dual variable y_i initialized as 0 is used as a proxy in round i. We increment $x^{(r)}$ until the arriving violating constraint i is satisfied *by a factor of 2*. This guarantees that at least one variable $x_j^{(r)}$ is doubled thus ensures that the algorithm only encounters polynomially many violating constraints.

[1]The same initialization is used for the same reason for online covering LPs with box constraints and online covering SDPs with and without box constraints.

In each phase r, each *iteration* ends in one of the following three cases: (1) the arriving constraint i is satisfied by a factor of 2, (2) there exists a variable $x_j^{(r)}$ that reaches the advice value x'_j, or (3) the objective $c^T x^{(r)}$ reaches $\alpha(r)$. The coefficients D_j and B_j for fitting boundary conditions are defined based on the value of $x_j^{(r)}$ in the end of the previous iteration, stored as $\bar{x}_j$. The indicator $\mathbf{1}_{x_j^{(r)}<x'_j}$ used for the growth rate, has the same value as $\mathbf{1}_{\bar{x}_j<x'_j}$ during an iteration.

The main algorithm that uses the phase scheme is presented in Algorithm 1. The continuous primal-dual approach in phase r and round i, used as a subroutine, is presented in Algorithm 2.

Algorithm 1 Phase Scheme for Algorithm 2

1: $r \leftarrow 1$, $\alpha(1) \leftarrow \min_{j\in[n]}\{c_j/a_{1j}\}$, $i_1 \leftarrow 1$. ▷ initialization for round 1
2: **for** each $j \in [n]$ **do**
3: $\quad x_j^{(r)} \leftarrow \min\{x'_j, \alpha(r)/(2nc_j)\}$.
4: **for** arriving covering constraint $A_i x \geq 1$ **do** ▷ $i = 1, 2, ..., m$ for an unknown m
5: $\quad$ Run Algorithm 2.
6: $\quad$ **if** $c^T x^{(r)} \geq \alpha(r)$ **then** ▷ start a new phase
7: $\quad\quad r \leftarrow r+1$, $\alpha(r) \leftarrow 2\alpha(r-1)$, $i_r \leftarrow i$.
8: $\quad\quad$ **for** each $j \in [n]$ **do**
9: $\quad\quad\quad x_j^{(r)} \leftarrow \min\{x'_j, \alpha(r)/(2nc_j)\}$.
10: $\quad\quad$ Go to line 5.
11: $\quad$ **for** each $j \in [n]$ **do**
12: $\quad\quad x_j \leftarrow \max_{\ell\in[r]}\{x_j^{(\ell)}\}$. ▷ this is the solution returned in each round i

We note that when we start a new phase r from the violating constraint i_r, y_{i_r} is set to zero. In phase r and round i, we are actually considering the following covering and packing LPs:

$$\text{minimize } c^T x^{(r)} \text{ over } x^{(r)} \in \mathbb{R}^n_{\geq 0} \text{ subject to } A_k x^{(r)} \geq 1 \text{ for } k = i_r, ..., i \tag{10}$$

and

$$\text{maximize } \mathbf{1}^T y \text{ over } y_k \geq 0 \text{ for } k = i_r, ..., i \text{ subject to } \sum_{k=i_r}^{i} a_{kj} y_k \leq c_j \text{ for } j \in [n]. \tag{11}$$

Although the augmentation is in a continuous fashion, it is not hard to implement it in a discrete way for any desired precision by binary search. The approach of satisfying each arriving violating constraint by a factor of 2 guarantees that the number of iterations is polynomially upper-bounded. This implies efficient applications on problems that generate covering LPs with exponentially many or unbounded number of constraints, where violating constraints are retrieved by a separation oracle. The performance of Algorithm 1 is stated in Theorem 2.1, the formal version of Theorem 1.3.

Theorem 2.1. *For the learning-augmented online covering LP problem, there exists an online algorithm that generates x such that*

$$c^T x \leq \min\left\{O\left(\frac{1}{1-\lambda}\right) c^T x' + O(\log(\kappa n))\mathit{OPT}, O\left(\log\frac{\kappa n}{\lambda}\right)\mathit{OPT}\right\}$$

and encounters $O(n(\log\frac{c^T x}{\alpha(1)})(\log n + \log c^T x + \log\beta))$ violating constraints. If x' is feasible for LP (1), then $c^T x \leq O\left(\frac{1}{1-\lambda}\right) c^T x'$. Here, $\kappa := \max_{j\in[n]}\{a_j^{\max}/a_j^{\min}\}$, $\beta = \max_{j\in[n]}\{a_j^{\max}/c_j\}$, $a_j^{\max} := \max_{i\in[m]}\{a_{ij} \mid a_{ij} > 0\}$, and $a_j^{\min} := \min_{i\in[m]}\{a_{ij} \mid a_{ij} > 0\}$.

Algorithm 2 PDLA Online Covering in Phase r and Round i

Input: $x^{(r)}$: current solution, $\alpha(r)$: estimate for OPT, A_i: current row, i_r: starting round of phase r, y_k for $k = i_r, ..., i-1$, λ: the confidence parameter, and x': the advice for x.
Output: Updated $x^{(r)}$ and y_i.

1: $y_i \leftarrow 0$. ▷ the packing variable y_i is used for the analysis
2: $\bar{x} \leftarrow x^{(r)}$. ▷ $\bar{x}$ is the value of $x^{(r)}$ at the end of the previous iteration
3: **for** $j \in [n]$ **do**
4: **if** $A_i x' \geq 1$ **then**
5: $D_j \leftarrow \frac{\lambda}{A_i \mathbf{1}} + \frac{(1-\lambda)x'_j \mathbf{1}_{\bar{x}_j < x'_j}}{A_i x'_c}$,
6: **else**
7: $D_j \leftarrow \frac{1}{A_i \mathbf{1}}$.
8: $Y_j^{(i-1)} \leftarrow \sum_{k=i_r}^{i-1} a_{kj} y_k$. ▷ if phase r just started, then $i = i_r$, so $Y_j^{(i-1)} \leftarrow 0$
9: $B_j \leftarrow \frac{\bar{x}_j + D_j}{\exp\left(\left(Y_j^{(i-1)} + a_{ij} y_i\right)/c_j\right)}$.
10: **if** $A_i x^{(r)} < 1$ **then**
11: **while** $A_i x^{(r)} < 2$ **do**
12: Increase y_i continuously.
13: **for** each $j \in [n]$ **do**
14: Increase $x_j^{(r)}$ simultaneously by

$$x_j^{(r)} \leftarrow B_j \exp\left(\frac{Y_j^{(i-1)} + a_{ij} y_i}{c_j}\right) - D_j.$$

15: **if** any $x_j^{(r)}$ reaches x'_j **then**
16: Break and go to line 2.
17: **if** $c^T x^{(r)} \geq \alpha(r)$ **then**
18: Break and return.

Proof. Let $P(r) = c^T x^{(r)}$ and $D(r) = \mathbf{1}^T y$ be the objective value of the primal and the dual in phase r, respectively. We use Algorithm 1 and first show robustness, i.e., $c^T x \leq O\left(\log \frac{\kappa n}{\lambda}\right) \mathsf{OPT}$, then show consistency, i.e., $c^T x \leq O\left(\frac{1}{1-\lambda}\right) c^T x' + O(\log(\kappa n))\mathsf{OPT}$. Finally, we show that only $O(n(\log \frac{c^T x}{\alpha(1)})(\log n + \log c^T x + \log \beta))$ violating constraints are encountered.

Robustness. To show that $c^T x \leq O\left(\log \frac{\kappa n}{\lambda}\right) \mathsf{OPT}$, we prove the following five claims:

(i) x is feasible for LP (1).

(ii) For each *finished* phase r, $\alpha(r) \leq 6D(r)$.

(iii) $y/\Theta\left(\log \frac{\kappa n}{\lambda}\right)$ is feasible for LP (11) in each phase.

(iv) The sum of the covering objective generated from phase 1 to r is at most $2\alpha(r)$.

(v) Let r' be the last phase, then the covering objective $c^T x \leq 2\alpha(r')$.

Equipped with these claims and weak duality, we have that

$$c^T x \leq \Theta(1)\alpha(r') \leq \Theta(1) D(r') = O\left(\log \frac{\kappa n}{\lambda}\right) \mathsf{OPT}.$$

Proof of (i). We prove that $x^{(r)}$ is feasible in phase r by showing that the growing function in Algorithm 2 line 14 increments $x^{(r)}$ in a continuous manner. In the beginning of an iteration in round i, we have that

$$x_j^{(r)} = B_j \exp\left(\frac{Y_j^{(i-1)} + a_{ij}y_i}{c_j}\right) - D_j = \frac{\bar{x}_j + D_j}{\exp\left(\left(Y_j^{(i-1)} + a_{ij}y_i\right)/c_j\right)} \exp\left(\frac{Y_j^{(i-1)} + a_{ij}y_i}{c_j}\right) - D_j = \bar{x}_j.$$

By following Algorithm 2, we have that $x^{(r)}$ is feasible for LP (10) since we terminate the while loop at line 11 when $A_i x^{(r)} = 2 > 1$. Then, as x is the coordinate-wise maximum of $\{x^{(r)}\}$, x must be feasible for LP (1). □

Proof of (ii). In the beginning of phase r, $x_j^{(r)} = \min\{x'_j, \alpha(r)/(2nc_j)\}$, so $P(r)$ is initially at most $\alpha(r)/2$. The total increase of $P(r)$ is at least $\alpha(r)/2$ as $P(r) \geq \alpha(r)$ when phase r ends. Therefore, it suffices to show that in round i,

$$\frac{\partial P(r)}{\partial y_i} \leq 3\frac{\partial D(r)}{\partial y_i}.$$

By considering the partial derivative of $P(r)$ with respect to y_i, when $A_i x' \geq 1$, we have that

$$\begin{aligned}
\frac{\partial P(r)}{\partial y_i} &= \sum_{j=1}^{n} c_j \frac{\partial x_j^{(r)}}{\partial y_i} \\
&= \sum_{j=1}^{n} a_{ij}c_j \left(\frac{B_j}{c_j} \exp\left(\frac{Y_j^{(i-1)} + a_{ij}y_i}{c_j}\right)\right) \\
&= \sum_{j=1}^{n} a_{ij} \left(B_j \exp\left(\frac{Y_j^{(i-1)} + a_{ij}y_i}{c_j}\right) - D_j + D_j\right) \\
&= \sum_{j=1}^{n} a_{ij} \left(x_j^{(r)} + \frac{\lambda}{A_i\mathbf{1}} + \frac{(1-\lambda)x'_j \mathbf{1}_{x_j^{(r)} < x'_j}}{A_i x'_c}\right) \\
&\leq 2 + \lambda + (1-\lambda) = 3 = 3\frac{\partial D(r)}{\partial y_i}
\end{aligned}$$

where the last inequality is due to the fact that $\sum_{j=1}^{n} a_{ij}x_j^{(r)} < 2$ and (9). The same result can be obtained when $A_i x' < 1$ by regarding λ as 1. □

Proof of (iii). We show that $y/\Theta\left(\log \frac{\kappa n}{\lambda}\right)$ is feasible for LP (11) in the last round ℓ_r of phase r. The argument applies to any round prior to ℓ_r. We recall that within a phase r, each iteration ends in one of the following two cases: (1) the arriving constraint i is satisfied by a factor of 2, or (2) there exists a variable $x_j^{(r)}$ that reaches the advice value x'_j. Suppose there are $t \leq \ell_r - i_r + 1 + n$ iterations in phase r since there are $\ell_r - i_r + 1$ covering constraints and n variables advised. In iteration $p \leq t$, let $x_j^{(r,p)}$, $B_j^{(p)}$, and $D_j^{(p)}$ be the value of $x_j^{(r)}$, B_j, and D_j in the end of iteration p, respectively. Additionally, let $i^{(p)}$ denote the corresponding round in which iteration p occurs and $\hat{Y}_j^{(p)} = \sum_{k=i_r}^{i^{(p)}} a_{kj}y_k$ be the accumulative weighted dual variable sum in the end of iteration p with respect to coordinate j in the primal. Note that we are only incrementing $y_{i^{(p)}}$ in iteration p. In the beginning of phase r, $Y^{(i_r)} = \hat{Y}^{(0)}$ is a zero vector; in the end of the last round ℓ_r, $\hat{Y}^{(t)} = Y^{(\ell_r)}$. We have that

$$x_j^{(r,t)} = B_j^{(t)} \exp\left(\frac{\hat{Y}_j^{(t)}}{c_j}\right) - D_j^{(t)} \implies \frac{\hat{Y}_j^{(t)}}{c_j} = \ln \frac{x_j^{(r,t)} + D_j^{(t)}}{B_j^{(t)}}.$$

Then,

$$\begin{aligned}\frac{\hat{Y}_j^{(t)}}{c_j} &= \ln \frac{x_j^{(r,t)} + D_j^{(t)}}{B_j^{(t)}} = \ln \left(\frac{x_j^{(r,t)} + D_j^{(t)}}{x_j^{(r,t-1)} + D_j^{(t)}} \exp\left(\frac{\hat{Y}_j^{(t-1)}}{c_j} \right) \right) \\ &= \frac{\hat{Y}_j^{(t-1)}}{c_j} + \ln \frac{x_j^{(r,t)} + D_j^{(t)}}{x_j^{(r,t-1)} + D_j^{(t)}} \\ &= \dots \\ &= \frac{\hat{Y}_j^{(0)}}{c_j} + \sum_{p=1}^{t} \ln \frac{x_j^{(r,p)} + D_j^{(p)}}{x_j^{(r,p-1)} + D_j^{(p)}} \\ &= \ln \prod_{p=1}^{t} \frac{x_j^{(r,p)} + D_j^{(p)}}{x_j^{(r,p-1)} + D_j^{(p)}}.\end{aligned}$$

From here, we use the following claims.

Claim 2.2. *For all scalars $a \geq b > 0$ and $c_1 \geq c_2 \geq 0$,*

$$\frac{a + c_1}{b + c_1} \leq \frac{a + c_2}{b + c_2}.$$

Claim 2.3. $x_j^{(r)} \leq 2/a_j^{\min}$.

Proof. Suppose $x_j^{(r)} > 2/a_j^{\min}$, then by definition of $a_j^{\min}$, if $x_j^{(r)} > 2/a_j^{\min}$ in the current round, then it contradicts to the terminating condition in Algorithm 2 line 11 since $A_i x^{(r)} \geq a_j^{\min} x_j^{(r)} > 2$; if $x_j^{(r)} > 2/a_j^{\min}$ in an earlier round of phase r, then it contradicts to the condition that the current round is violated since $A_i x^{(r)} \geq a_j^{\min} x_j^{(r)} > 2$. □

Recall that $\kappa := \max_{j \in [n]} \{a_j^{\max}/a_j^{\min}\}$, $a_j^{\max} := \max_{i \in [m]} \{a_{ij} \mid a_{ij} > 0\}$, and $a_j^{\min} := \min_{i \in [m]} \{a_{ij} \mid a_{ij} > 0\}$. Notice that $D_j^{(p)} \geq \frac{\lambda}{a_j^{\max} n}$ for all $p \leq t$. Let $D_j^{min} := \min_{p \in [t]} \{D_j^{(p)}\} \geq \frac{\lambda}{a_j^{\max} n}$. By Claim 2.2, we have that

$$\begin{aligned}\frac{\hat{Y}_j^{(t)}}{c_j} &= \ln \prod_{p=1}^{t} \frac{x_j^{(r,p)} + D_j^{(p)}}{x_j^{(r,p-1)} + D_j^{(p)}} \leq \ln \prod_{p=1}^{t} \frac{x_j^{(r,p)} + D_j^{min}}{x_j^{(r,p-1)} + D_j^{min}} \\ &= \ln \frac{x_j^{(r,t)} + D_j^{min}}{x_j^{(r,0)} + D_j^{min}} \leq \ln \left(1 + \frac{2}{a_j^{\min} D_j^{min}} \right) = O\left(\log \frac{\kappa n}{\lambda} \right)\end{aligned}$$

where the last inequality is by Claim 2.3 which implies that $x_j^{(r,t)} \leq 2/a_j^{\min}$. □

Proof of (iv). The sum of the covering objective generated from phase 1 to r is at most

$$\sum_{k=1}^{r} \alpha(k) = \sum_{k=1}^{r} \frac{\alpha(r)}{2^{k-r}} \leq 2\alpha(r).$$

□

Proof of (v). In the last phase r', x is feasible because it is the coordinate-wise maximum of $\{x^{(r)}\}_{r \in [r']}$. We have

$$c^T x = \sum_{j=1}^{n} c_j x_j \leq \sum_{j=1}^{n} \sum_{r=1}^{r'} c_j x_j^{(r)} = \sum_{r=1}^{r'} \left(\sum_{j=1}^{n} c_j x_j^{(r)} \right) \leq \sum_{r=1}^{r'} \alpha(r) \leq 2\alpha(r')$$

where the first inequality holds because $x_j = \max_r\{x_j^{(r)}\} \le \sum_r x_j^{(r)}$, the second inequality is by the fact that the covering objective $\sum_j c_j x_j^{(r)}$ cannot exceed the estimated lower bound $\alpha(r)$, while the last inequality is by (iv). □

Consistency. We then show that $c^T x \le O\left(\frac{1}{1-\lambda}\right) c^T x' + O(\log(\kappa n))\mathsf{OPT}$. Suppose Algorithm 2 is in phase r with constraint i arriving. If $A_i x' < 1$, then the change of $c^T x$ simply follows (ii) and (iii) by regarding λ as 1, so $c^T x \le O(\log(\kappa n))\mathsf{OPT}$. For the more interesting case when $A_i x' \ge 1$, we decompose $\frac{\partial P(r)}{\partial y_i} = \frac{\partial P_c}{\partial y_i} + \frac{\partial P_u}{\partial y_i}$, where P_c is the component of the primal objective due to the advice and P_u is the component of the primal objective due to the online algorithm. We have that the rate of change is credited to ∂P_c if $x_j^{(r)} < x'_j$ and the rate of change is credited to ∂P_u otherwise, if $x_j^{(r)} \ge x'_j$. It suffices to check the rate of change since $x_j^{(r)}$ is initialized to $\min\{x'_j, \alpha(r)/(2nc_j)\}$, i.e., P_c is non-negative and $P_u = 0$ in the beginning of phase r.[2] In particular, in round i,

$$\frac{\partial P_c}{\partial y_i} = \sum_{j\in[n]:x_j^{(r)}<x'_j} a_{ij}\left(x_j^{(r)} + \frac{\lambda}{A_i\mathbf{1}} + \frac{(1-\lambda)x'_j\mathbf{1}_{x_j^{(r)}<x'_j}}{A_i x'_c}\right) \ge 0 + \frac{a_j^{\min}\lambda}{a_j^{\max}n} + (1-\lambda),$$

$$\frac{\partial P_u}{\partial y_i} = \sum_{j\in[n]:x_j^{(r)}\ge x'_j} a_{ij}\left(x_j^{(r)} + \frac{\lambda}{A_i\mathbf{1}} + \frac{(1-\lambda)x'_j\mathbf{1}_{x_j^{(r)}<x'_j}}{A_i x'_c}\right) \le 2 + \lambda + 0.$$

Thus we have $\frac{\partial P_u}{\partial y_i} \le \frac{2+\lambda}{a_j^{\min}\lambda/\left(a_j^{\max}n\right)+1-\lambda} \cdot \frac{\partial P_c}{\partial y_i}$, so that

$$\frac{\partial P(r)}{\partial y_i} \le \left(1 + \frac{2+\lambda}{\frac{a_j^{\min}\lambda}{a_j^{\max}n} + 1 - \lambda}\right)\frac{\partial P_c}{\partial y_i} = O\left(\frac{1}{1-\lambda}\right)\frac{\partial P_c}{\partial y_i}.$$

We note that if x' is feasible, then $A_i x' \ge 1$ for all $i \in [m]$, so $c^T x \le O\left(\frac{1}{1-\lambda}\right) c^T x'$.

Bounding the number of violating constraints. Finally, we show that Algorithm 1 encounters $O(n(\log\frac{c^T x}{\alpha(1)})(\log n + \log c^T x + \log\beta))$ violating constraints. We first show that there are $O(\log(c^T x/\alpha(1)))$ phases. The estimated lower bound α doubles when we start a new phase. Suppose there are r' phases, then $\alpha(1)\cdot 2^{r'-1} = O(c^T x)$. This implies that $r' = O(\log(c^T x/\alpha(1)))$.

In each phase r, when a violating constraint i just arrived, we increment $x^{(r)}$ until the constraint is satisfied by a factor of 2. One of the following two cases must hold after updating $x^{(r)}$: (1) there exists a *large* variable $x_j^{(r)} \ge 1/(2na_j^{\max})$ that is updated to at least $3x_j^{(r)}/2$, or (2) there exists a *small* variable $x_j^{(r)} < 1/(2na_j^{\max})$ that becomes large, i.e., $x_j^{(r)}$ is updated to at least $1/(2na_j^{\max})$. Let L and S be the set of large and small variable subscript labels before the violating constraint i arrives, respectively, and $\hat{x}_j^{(r)}$ be the value of $x_j^{(r)}$ after the update. If none of these two cases holds, then

$$\sum_{j=1}^n a_{ij}\hat{x}_j^{(r)} < \frac{3}{2}\sum_{j\in L} a_{ij}x_j^{(r)} + \sum_{j\in S}\frac{a_{ij}}{2na_j^{\max}} < \frac{3}{2} + \frac{1}{2} = 2$$

where the second inequality is by the fact that constraint i is violated and $a_{ij} \le a_j^{\max}$. This implies that constraint i is not satisfied by a factor of 2 after the update, a contradiction.

[2] The same argument is used for proving the consistency in Theorems 2.4, 3.1, and 3.2.

Suppose $x_j^{(r)}$ has been updated t times since it was large in phase r, then

$$\frac{c_j}{2na_j^{\max}}\left(\frac{3}{2}\right)^t = O(c^T x)$$

which implies that $t = O(\log n + \log c^T x + \log(a_j^{\max}/c_j)) = O(\log n + \log c^T x + \log \beta)$.

There are n variables, each variable can be updated from small to large once and updated t times by a factor of $3/2$ since it was large in each phase. Hence, Algorithm 1 encounters $O(n(\log \frac{c^T x}{\alpha(1)})(\log n + \log c^T x + \log \beta))$ violating covering constraints. □

2.1 Adding Box Constraints

We recall that the covering LP (7) with box constraints is the following.

$$\text{minimize } c^T x \text{ over } x \in [0,1]^n \text{ subject to } Ax \geq \mathbf{1}.$$

Our PDLA algorithm simultaneously considers the dual packing LP:

$$\text{maximize } \mathbf{1}^T y - \mathbf{1}^T z \text{ over } y \in \mathbb{R}^m_{\geq 0} \text{ and } z \in \mathbb{R}^n_{\geq 0} \text{ subject to } A^T y - z \leq c. \tag{12}$$

We recall that we assume that the advice $x' \in [0,1]^n$.

We use the guess-and-double approach similar to Algorithm 1 and 2, with a tweak of maintaining the set T, which denotes the subscript indices of the x variables that are *tight*. In phase r, upon the arrival of a violating constraint i, we increment $x_j^{(r)}$ in terms of an exponential function of y_i and z_j subject to $x_j^{(r)} \leq 1$. Once $x_j^{(r)} = 1$, we add j to the tight set T and stop incrementing $x_j^{(r)}$, but we still increment y_i continuously and z_j with rate $a_{ij}y_i$ in order to maintain dual feasibility. In the beginning of each phase, z is a zero vector and T is an empty set. $z_j = 0$ whenever $j \in [n] \setminus T$.

Subject to $x_j^{(r)} \leq 1$, $x_j^{(r)}$ is increased until the cost *outside of the tight set* exceeds the *remaining capacity* by a factor of 2. More specifically, when we have a violating constraint i, we have that $\sum_{j\in[n]\setminus T} a_{ij}x_j^{(r)} < 1 - \sum_{j\in T} a_{ij}$, and we increment $x_j^{(r)}$ until $\sum_{j\in[n]\setminus T} a_{ij}x_j^{(r)} \geq 2\left(1 - \sum_{j\in T} a_{ij}\right)$. Let A'_i denote the vector with entries of A_i by considering only the coordinates that are not tight. The entry is zero if the coordinate is tight. More formally,

$$a'_{ij} = \begin{cases} a_{ij} & \text{if } x_j < 1, \text{ i.e., } j \in [n] \setminus T, \\ 0 & \text{if } x_j = 1, \text{ i.e., } j \in T, \end{cases}$$

where a'_{ij} denotes the j-th coordinate entry of A'_i.

If the advice adequately covers constraint i, i.e., $A'_i x' \geq 1$, we increase the variables $x_j^{(r)}$ with growth rate

$$\frac{a_{ij}}{c_j}\left(x_j^{(r)} + \left(\frac{\lambda}{A'_i\mathbf{1}} + \frac{(1-\lambda)x'_j\mathbf{1}_{x_j^{(r)}<x'_j}}{A'_i x'_c}\right)\left(1 - \sum_{j\in T} a_{ij}\right)\right).$$

Alternatively if the advice does not cover the constraint enough, i.e., $A'_i x' < 1$, we increase $x_j^{(r)}$ with growth rate

$$\frac{a_{ij}}{c_j}\left(x_j^{(r)} + \frac{1 - \sum_{j\in S} a_{ij}}{A'_i\mathbf{1}}\right).$$

We note that we increment $x^{(r)}$ only when $1 - \sum_{j\in T} a_{ij} > 0$, since otherwise the constraint $A_i x^{(r)} \geq \sum_{j\in T} a_{ij} \geq 1$ is already satisfied and there is nothing to be updated. Similar to (9), we have that

$$\sum_{j\in[n]\setminus T} \frac{a_{ij}}{A'_i\mathbf{1}} = 1 \text{ and } \sum_{j\in[n]\setminus T} \frac{a_{ij}x'_j\mathbf{1}_{x_j^{(r)}<x'_j}}{A'_i x'_c} = 1. \tag{13}$$

Each iteration now ends in one of the following four cases: (1) the arriving constraint i induces a cost outside of the tight set that exceeds the remaining capacity by a factor of 2, (2) there exists a variable $x_j^{(r)}$ that reaches the advice value x'_j, (3) there exists a variable $x_j^{(r)}$ that reaches 1, or (4) the objective $c^T x^{(r)}$ reaches $\alpha(r)$. The coefficients D_j and B_j for fitting boundary conditions are defined based on the value of $x_j^{(r)}$ in the end of the previous iteration, stored as $\bar{x}_j$. Again, the indicator value used for the growth rate, $\mathbf{1}_{x_j^{(r)}<x'_j}$, has the same value as $\mathbf{1}_{\bar{x}_j<x'_j}$ during an iteration.

The main algorithm that uses the phase scheme is presented in Algorithm 3. The continuous primal-dual approach in phase r and round i, used as a subroutine, is presented in Algorithm 4.

Algorithm 3 Phase Scheme for Algorithm 4

$r \leftarrow 1$, $\alpha(1) \leftarrow \min_{j\in[n]}\{c_j/a_{1j}\}$, $T \leftarrow \emptyset$, $i_1 \leftarrow 1$. ▷ initialization for round 1
for each $j \in [n]$ **do**
  $x_j^{(r)} \leftarrow \min\{x'_j, \alpha(r)/(2nc_j)\}$, $z_j \leftarrow 0$.
for arriving covering constraint $A_i x \geq 1$ **do** ▷ $i = 1, 2, ..., m$ for an unknown m
  Run Algorithm 4.
  if $c^T x^{(r)} \geq \alpha(r)$ **then** ▷ start a new phase
    $r \leftarrow r+1$, $\alpha(r) \leftarrow 2\alpha(r-1)$, $T \leftarrow \emptyset$, $i_r \leftarrow i$.
    for each $j \in [n]$ **do**
      $x_j^{(r)} \leftarrow \min\{x'_j, \alpha(r)/(2nc_j)\}$, $z_j \leftarrow 0$.
    Go to line 5.
  for each $j \in [n]$ **do**
    $x_j \leftarrow \max_{\ell\in[r]}\{x_j^{(\ell)}\}$. ▷ this is the solution returned in each round i

We note that when we start a new phase r from the violating constraint i_r, y_{i_r} is set to zero and z is set to a zero vector. In phase r and round i, we are actually considering the following covering and packing LPs with box constraints:

$$\text{minimize } c^T x^{(r)} \text{ over } x^{(r)} \in [0,1]^n \text{ subject to } A_k x^{(r)} \geq 1 \text{ for } k = i_r, ..., i. \tag{14}$$

and

$$\text{maximize } \mathbf{1}^T y - \mathbf{1}^T z \text{ over } y_k \geq 0 \text{ for } k = i_r, ..., i \text{ and } z \in \mathbb{R}^n_{\geq 0} \text{ subject to } \sum_{k=i_r}^{i} a_{kj} y_k - z_j \leq c_j \text{ for } j \in [n]. \tag{15}$$

In Algorithm 4 line 16, when $j \in T$, $x_j^{(r)}$ remains unchanged, since z_j is increased with rate $a_{ij}y_i$. This ensures that approximate dual feasibility is maintained when $x_j^{(r)}$ is tight. The performance of Algorithm 3 is stated in Theorem 2.4, the formal version of Theorem 1.5.

Theorem 2.4. *For the learning-augmented online covering LP problem with box constraints, there exists an online algorithm that generates x such that*

$$c^T x \leq \min\left\{ O\left(\frac{1}{1-\lambda}\right) c^T x' + O(\log s)\mathit{OPT}, O\left(\log \frac{s}{\lambda}\right)\mathit{OPT} \right\}$$

and encounters $O(n \log s \log \frac{c^T x}{\alpha(1)})$ violating constraints. If x' is feasible for LP (7), then $c^T x \leq O\left(\frac{1}{1-\lambda}\right) c^T x'$. Here,

$$s := \max_{i\in[m], T\subseteq[n]} \left\{ \frac{\sum_{j\in[n]\setminus T} a_{ij}}{1 - \sum_{j\in T} a_{ij}} \mid 1 - \sum_{j\in T} a_{ij} > 0 \right\}.$$

Algorithm 4 PDLA Online Covering with Box Constraints in Phase r and Round i

Input: $x^{(r)}$: current solution, $\alpha(r)$: estimate for OPT, A_i: current row, i_r: starting round of phase r, y_k for $k = i_r, ..., i-1$, z: dual variable vector, T: tight variable set, λ: the confidence parameter, and x': the advice for x.

Output: Updated $x^{(r)}$, y_i, and z.

1: $y_i \leftarrow 0$. ▷ the packing variable y_i is used for the analysis

2: $\bar{x} \leftarrow x^{(r)}$. ▷ $\bar{x}$ is the value of $x^{(r)}$ at the end of the previous iteration

3: **for** $j \in [n]$ **do**

4: **if** $A_i x' \geq 1$ **then**

5: $D_j \leftarrow \left(\frac{\lambda}{A_i'\mathbf{1}} + \frac{(1-\lambda)x_j' \mathbf{1}_{\bar{x}_j < x_j'}}{A_i' x_c'}\right)\left(1 - \sum_{j \in T} a_{ij}\right)$,

6: **else**

7: $D_j \leftarrow \frac{1 - \sum_{j \in T} a_{ij}}{A_i'\mathbf{1}}$.

8: $Y_j^{(i-1)} \leftarrow \sum_{k=i_r}^{i-1} a_{kj} y_k$. ▷ if phase r just started, then $i = i_r$, so $Y_j^{(i-1)} \leftarrow 0$

9: $B_j \leftarrow \frac{\bar{x}_j + D_j}{\exp\left(\left(Y_j^{(i-1)} + a_{ij} y_i - z_j\right)/c_j\right)}$.

10: **if** $A_i x^{(r)} < 1$ **then**

11: **while** $T \neq [n]$ and $\sum_{j \in [n] \setminus T} a_{ij} x_j^{(r)} < 2\left(1 - \sum_{j \in T} a_{ij}\right)$ **do**

12: Increase y_i continuously.

13: **for** each $j \in [n]$ **do**

14: **if** $j \in T$ **then**

15: Increase z_j with rate $a_{ij} y_i$.

16: Update $x_j^{(r)}$ simultaneously by

$$x_j^{(r)} \leftarrow \min\left\{1, B_j \exp\left(\frac{Y_j^{(i-1)} + a_{ij} y_i - z_j}{c_j}\right) - D_j\right\}.$$

17: **if** any $x_j^{(r)} = x_j' < 1$ **then**

18: Break and go to line 2.

19: **if** any $x_j^{(r)} = 1$ for $j \notin T$ **then**

20: Add j to T and go to line 2.

21: **if** $c^T x^{(r)} \geq \alpha(r)$ **then**

22: Break and return.

23: **if** $T = [n]$ and $A_i x^{(r)} < 1$ **then**

24: **return** no feasible solution.

Proof. Let $P(r) = c^T x^{(r)}$ and $D(r) = \mathbf{1}^T y - \mathbf{1}^T z$ be the objective value of the primal and the dual in phase r, respectively. We use Algorithm 3 and first show robustness, i.e., $c^T x \leq O\left(\log \frac{s}{\lambda}\right)\mathsf{OPT}$, then show consistency, i.e., $c^T x \leq O\left(\frac{1}{1-\lambda}\right) c^T x' + O(\log s)\mathsf{OPT}$. Finally, we show that only $O(n \log s \log \frac{c^T x}{\alpha(1)})$ violating constraints are encountered.

Robustness. To show that $c^T x \leq O\left(\log \frac{s}{\lambda}\right)\mathsf{OPT}$, we prove the following five claims:

(i) If line 24 in Algorithm 4 is not encountered, then x is feasible for LP (7).

(ii) For each *finished* phase r, $\alpha(r) \leq 6D(r)$.

(iii) $(y, z)/\Theta\left(\log \frac{s}{\lambda}\right)$ is feasible for LP (15) in each phase.

(iv) The sum of the covering objective generated from phase 1 to r is at most $2\alpha(r)$.

(v) Let r' be the last phase, then the covering objective $c^T x \le 2\alpha(r')$.

The proofs of (iv) and (v) are the same as the ones in Theorem 2.1, hence omitted. Equipped with these claims and weak duality, we have that

$$c^T x \le \Theta(1)\alpha(r') \le \Theta(1)D(r') = O\left(\log \frac{s}{\lambda}\right)\mathsf{OPT}.$$

Proof of (i). We prove that $x^{(r)}$ is feasible in phase r by showing that the growing functions in Algorithm 4 line 16 increments $x^{(r)}$ in a continuous manner. In the beginning of an iteration in round i, we have that

$$x_j^{(r)} = B_j \exp\left(\frac{Y_j^{(i-1)} + a_{ij}y_i}{c_j}\right) - D_j = \frac{\bar{x}_j + D_j}{\exp\left(\left(Y_j^{(i-1)} + a_{ij}y_i\right)/c_j\right)} \exp\left(\frac{Y_j^{(i-1)} + a_{ij}y_i}{c_j}\right) - D_j = \bar{x}_j.$$

By following Algorithm 4, we have that $x^{(r)}$ is feasible for LP (14) since we terminate the while loop at line 11 when the cost outside of the tight set exceeds the remaining capacity by a factor of 2. Then, as x is the coordinate-wise maximum of $\{x^{(r)}\}$, capped at 1, x must be feasible for LP (7). □

Proof of (ii). In the beginning of phase r, $x_j^{(r)} = \min\{x'_j, \alpha(r)/(2nc_j)\}$, so $P(r)$ is initially at most $\alpha(r)/2$. The total increase of $P(r)$ is at least $\alpha(r)/2$ as $P(r) \ge \alpha(r)$ when phase r ends. Therefore, it suffices to show that

$$\frac{\partial P(r)}{\partial y_i} \le 3\frac{\partial D(r)}{\partial y_i}.$$

Recall that T is the set of the tight x indices and the z_j variables in T are increasing with rate $a_{ij}y_i$, we have that

$$\frac{\partial D(r)}{\partial y_i} = 1 - \sum_{j\in T} a_{ij}.$$

When y_i is increasing, this partial derivative is always non-negative due to the condition in Algorithm 4 line 11. Namely, at the point when $x_j^{(r)}$ is tight, we add j to T, and this may make the partial derivative change from non-negative to negative. In this case, we have that $2(1-\sum_{j\in T} a_{ij}) < 0$ so the condition automatically fails. From here, when $A_i x' \ge 1$, we have that

$$\begin{aligned}
\frac{\partial P(r)}{\partial y_i} &= \sum_{j\in[n]\setminus T} c_j \frac{\partial x_j^{(r)}}{\partial y_i} \\
&= \sum_{j\in[n]\setminus T} a_{ij}c_j \left(\frac{B_j}{c_j}\exp\left(\frac{Y_j^{(i-1)} + a_{ij}y_i - z_j}{c_j}\right)\right) \\
&= \sum_{j\in[n]\setminus T} a_{ij}\left(B_j \exp\left(\frac{Y_j^{(i-1)} + a_{ij}y_i - z_j}{c_j}\right) - D_j + D_j\right) \\
&= \sum_{j\in[n]\setminus T} a_{ij}\left(x_j^{(r)} + \left(\frac{\lambda}{A'_i\mathbf{1}} + \frac{(1-\lambda)x'_j\mathbf{1}_{\bar{x}_j<x'_j}}{A'_i x'_c}\right)\left(1 - \sum_{j\in T} a_{ij}\right)\right) \\
&\le 2\left(1 - \sum_{j\in T} a_{ij}\right) + (\lambda + (1-\lambda))\left(1 - \sum_{j\in T} a_{ij}\right) \\
&= 3\left(1 - \sum_{j\in T} a_{ij}\right) = 3\frac{\partial D(r)}{\partial y_i}
\end{aligned}$$

where the last inequality is due to the fact that $\sum_{j\in[n]\setminus T} a_{ij}x_j^{(r)} < 2\left(1-\sum_{j\in T} a_{ij}\right)$ and (13). The same result can be obtained when $A_i x' < 1$ by regarding λ as 1. □

Proof of (iii). We show that $(y,z)/\Theta\left(\log\frac{\kappa n}{\lambda}\right)$ is feasible for LP (15) in the last round ℓ_r of phase r. The argument applies to any round prior to ℓ_r. We recall that within a phase r, an iteration ends in one of the following three cases: (1) the arriving constraint i induces a cost outside of the tight set that exceeds the remaining capacity by a factor of 2, (2) there exists a variable $x_j^{(r)}$ that reaches the advice value x'_j, or (3) there exists a variable $x_j^{(r)}$ that reaches 1. Suppose there are $t \le \ell_r - i_r + 1 + 2n$ iterations in phase r since there are $\ell_r - i_r + 1$ covering constraints, n variables advised, and each variable is capped at 1 at most once. In iteration $p \le t$, let $x_j^{(r,p)}$, $B_j^{(p)}$, and $D_j^{(p)}$ be the value of $x_j^{(r)}$, B_j, and D_j in the end of iteration p, respectively. Additionally, let $i^{(p)}$ denote the corresponding round in which iteration p occurs and $\hat{Y}_j^{(p)} = \sum_{k=i_r}^{i^{(p)}} a_{kj}y_k - z_j$ be the accumulative weighted dual variable sum in the end of iteration p with respect to coordinate j in the primal. Note that we are only incrementing $y_{i^{(p)}}$ in iteration p and possibly incrementing z_j with rate $a_{i^{(p)}j}y_{i^{(p)}}$ if $j \in T$. If $j \in T$ in the beginning of iteration p, then $x_j^{(r,p-1)} = x_j^{(r,p)} = 1$ and $\hat{Y}_j^{(p-1)} = \hat{Y}_j^{(p)}$ because $a_{i^{(p)}j}y_j$ and z_j cancel out each other. The goal is to show that $\hat{Y}_j^{(t)}/c_j \le \Theta\left(\log\frac{s}{\lambda}\right)$.

By using a similar argument in Theorem 2.1 (iii), we can arrive at

$$\frac{\hat{Y}_j^{(t)}}{c_j} = \ln\prod_{p=1}^{t}\frac{x_j^{(r,p)}+D_j^{(p)}}{x_j^{(r,p-1)}+D_j^{(p)}} \le \ln\frac{x_j^{(r,t)}+D_j^{min}}{x_j^{(r,0)}+D_j^{min}}$$

where $D_j^{min} := \min_{p\in[t]}\{D_j^{(p)}\}$.

From here, notice that $x_j^{(r,t)} \le 1$, $x_j^{(r,0)} \ge 0$, and

$$D_j^{min} \ge \min_{\substack{i\in\{i_r,\dots,\ell_r\},T\subseteq[n]\\ 1-\sum_{j\in T}a_{ij}>0}}\left\{\frac{\lambda\left(1-\sum_{j\in T}a_{ij}\right)}{A'_i\mathbf{1}}\right\} \ge \min_{\substack{i\in\{i_r,\dots,\ell_r\},T\subseteq[n]\\ 1-\sum_{j\in T}a_{ij}>0}}\left\{\frac{\lambda\left(1-\sum_{j\in T}a_{ij}\right)}{\sum_{j\in[n]\setminus T}a_{ij}}\right\} \ge \frac{\lambda}{s}.$$

We thus have

$$\ln\frac{x_j^{(r,t)}+D_j^{min}}{x_j^{(r,0)}+D_j^{min}} \le \ln\left(1+\frac{1}{D_j^{min}}\right) = O\left(\log\frac{s}{\lambda}\right).$$

□

Consistency. We then show that $c^Tx \le O\left(\frac{1}{1-\lambda}\right)c^Tx' + O(\log s)\mathsf{OPT}$. Suppose Algorithm 4 is in phase r with constraint i arriving. If $A_ix' < 1$, then the change of c^Tx simply follows (ii) and (iii) by regarding λ as 1, so $c^Tx \le O(\log s)\mathsf{OPT}$. For the more interesting case when $A_ix' \ge 1$, we decompose $\frac{\partial P(r)}{\partial y_i} = \frac{\partial P_c}{\partial y_i} + \frac{\partial P_u}{\partial y_i}$, where P_c is the component of the primal objective due to the advice and P_u is the component of the primal objective due to the online algorithm. We have that the rate of change is credited to ∂P_c if $x_j^{(r)} < x'_j$ and the rate of change is credited to ∂P_u otherwise, if $x_j^{(r)} \ge x'_j$. In particular,

$$\frac{\partial P_c}{\partial y_i} = \sum_{j\in[n]\setminus T: x_j^{(r)}<x'_j} a_{ij}\left(x_j^{(r)} + \left(\frac{\lambda}{A'_i\mathbf{1}} + \frac{(1-\lambda)x'_j\mathbf{1}_{x_j^{(r)}<x'_j}}{A'_ix'_c}\right)\left(1-\sum_{j\in T}a_{ij}\right)\right) \ge 0 + \frac{\lambda}{s} + (1-\lambda),$$

$$\frac{\partial P_u}{\partial y_i} = \sum_{j\in[n]\setminus T: x_j^{(r)}\ge x'_j} a_{ij}\left(x_j^{(r)} + \left(\frac{\lambda}{A'_i\mathbf{1}} + \frac{(1-\lambda)x'_j\mathbf{1}_{x_j^{(r)}<x'_j}}{A'_ix'_c}\right)\left(1-\sum_{j\in T}a_{ij}\right)\right) \le 2 + \lambda + 0.$$

Thus we have $\frac{\partial P_u}{\partial y_i} \leq \frac{2+\lambda}{\lambda/s+1-\lambda} \cdot \frac{\partial P_c}{\partial y_i}$, so that

$$\frac{\partial P(r)}{\partial y_i} \leq \left(1 + \frac{2+\lambda}{\frac{\lambda}{s}+1-\lambda}\right)\frac{\partial P_c}{\partial y_i} = O\left(\frac{1}{1-\lambda}\right)\frac{\partial P_c}{\partial y_i}.$$

We note that if x' is feasible, then $A_i x' \geq 1$ for all $i \in [m]$, so $c^T x \leq O\left(\frac{1}{1-\lambda}\right) c^T x'$.

Bounding the number of violating constraints. We show that Algorithm 3 encounters $O(n \log s \log \frac{c^T x}{\alpha(1)})$ violating constraints. Using the same argument as in the proof of Theorem 2.4, we have that there are $O(\log(c^T x/\alpha(1)))$ phases.

In each phase r, when a violating constraint i just arrived, we have that $\sum_{j\in[n]\setminus T} a_{ij} x_j^{(r)} < 1 - \sum_{j\in T} a_{ij}$. We increment $x_j^{(r)}$'s where $j \in [n] \setminus T$ until $\sum_{j\in[n]\setminus T} a_{ij} x_j^{(r)} \geq 2\left(1 - \sum_{j\in T} a_{ij}\right)$. One of the following two cases must hold after updating $x^{(r)}$: (1) there exist a *large* variable $x_j^{(r)} \geq 1/(2s)$ that is updated to at least $3x_j^{(r)}/2$, or (2) there exists a *small* variable $x_j^{(r)} < 1/(2s)$ that becomes large, i.e., $x_j^{(r)}$ is updated to at least $1/(2s)$. Let L and S be the set of large and small variable subscript labels in $[n] \setminus T$ before the violating constraint i arrives, respectively, and $\hat{x}_j^{(r)}$ be the value of $x_j^{(r)}$ after the update. If none of these two cases holds, then

$$\sum_{j\in[n]\setminus T} a_{ij}\hat{x}_j^{(r)} < \frac{3}{2}\sum_{j\in L} a_{ij}x_j^{(r)} + \frac{1}{2s}\sum_{j\in S} a_{ij} < \frac{3}{2}\left(1 - \sum_{j\in T} a_{ij}\right) + \frac{\left(1-\sum_{k\in T} a_{ij}\right)}{2\sum_{k\in[n]\setminus T} a_{ij}}\sum_{j\in S} a_{ij} \leq 2\left(1 - \sum_{j\in T} a_{ij}\right)$$

where the second inequality is by the fact that constraint i is violated and the definition of the sparsity s, while the last inequality is by $S \subseteq [n] \setminus T$. This implies that the cost outside of the tight set does not exceed the remaining capacity by a factor of 2 after the update, a contradiction.

Suppose $x_j^{(r)}$ has been updated t times by a multiplicative factor of $3/2$ since it was large in phase r, then $t = O(\log s)$ since $\frac{1}{2s}(\frac{3}{2})^t \leq 1$.

There are n variables, each variable can be updated from small to large once and updated at most t times by a factor of $3/2$ since it was large in each phase. Hence, Algorithm 3 encounters $O(n \log s \log \frac{c^T x}{\alpha(1)})$ violating covering constraints. □

2.2 Applications

Our general framework on learning-augmented online covering LPs can be directly applied to online problems with predictions, including the online set cover problem and the online group Steiner tree problem on trees.

2.2.1 Online fractional set cover with fractional advice

In the online set cover problem introduced in [AAA+09], we are given offline n sets S_j where $j \in [n]$, each associated with a positive cost c_j. The elements arrive online one at a time. Upon the arrival of an element $i \in [m]$ where m can be unknown, the information of whether each set S_j contains element i is revealed. The goal is to irrevocably pick sets to cover all the elements that have arrived while minimizing the total cost. Let $F_i := \{j \in [n] \mid i \in S_j\}$ denote the subset of subscript indices with the corresponding sets containing element i. Let $d := \max_{i\in[m]}\{|F_i|\}$ denote the maximum number of sets that contain one specific element i, over all possible $i \in [m]$. An $O(\log m \log d)$-competitive algorithm was introduced in [AAA+09]. The online algorithm first finds an $O(\log d)$-competitive fractional solution for the following LP formulation:

$$\text{minimize } c^T x \text{ over } x \in [0,1]^n \text{ subject to } \sum_{j\in F_i} x_j \geq 1 \quad \forall i \in [m] \tag{16}$$

and then round by paying a factor of $O(\log m)$. The online fractional set cover problem considers LP (16), where each row is associated with an element that arrives and each column is associated with a set. The goal is to update x in a non-decreasing manner while minimizing the objective $c^T x$.

The learning-augmented problem was later introduced in [BMS20], where the input also includes a confidence parameter $\lambda \in [0,1]$ and an advice $A \subseteq [n]$ presented as a subset of subscript indices of the sets given offline. The advice A is feasible if $\cup_{j \in A} S_j = [m]$. Another way to present the advice is to describe it as a vector $x' \in \{0,1\}^n$, namely, set S_j is suggested by the advice if and only if $x'_j = 1$. Our framework therefore naturally extends to a more general setting when the advice can be fractional, i.e., $x' \in [0,1]^n$. Let OPT denote the optimal objective value for LP (16). We show that the online fractional set cover problem with fractional advice can be solved efficiently (independent of the number of elements) with the same approximation guarantee as [BMS20].

Corollary 2.5. *For the learning-augmented online fractional set cover problem with fractional advice, there exists an online algorithm that generates x and encounters polynomially many uncovered elements such that*

$$c^T x \leq \min \left\{ O\left(\frac{1}{1-\lambda}\right) c^T x' + O(\log d)\mathit{OPT}, O\left(\log \frac{d}{\lambda}\right)\mathit{OPT} \right\}.$$

If x' is feasible for LP (16), then $c^T x \leq O\left(\frac{1}{1-\lambda}\right) c^T x'$.

Proof. We use Theorem 2.4 and observe that the sparsity parameter $s = d$ since the entries of the constraint matrix is either 0 or 1. We note that the number of violating constraints encountered does not depend on m but depends on the cost c_j since $\alpha(1)$ depends on c_j. □

2.2.2 Online group Steiner tree on trees

We consider the learning-augmented online group Steiner tree problem on trees. We note that this problem generalizes other online problems including the online group Steiner tree problem on general graphs (by paying another logarithmic factor in the competitive ratio), the online multicast problem on trees, and the online non-metric facility location problem [AAA+06].

In the group Steiner tree problem on trees, we are given a weighted rooted tree $T = (V, E, r)$ and groups of vertices $V_1, V_2, ..., V_k \subseteq V$. Let $n = |V|$ denote the number of vertices. Each edge $e \in E$ is associated with a positive cost c_e. The goal is to find a minimum weighted rooted subtree $T' = (V', E', r)$ that contains at least one vertex from each group V_i. An $O(\log n \log k)$-approximation algorithm was presented in [GKR00] by considering the following LP formulation:

$$\text{minimize} \sum_{e \in E} c_e x_e \text{ over } x \in [0,1]^{|E|} \text{ subject to } x \text{ supports an } r \text{ to } V_i \text{ flow of value } 1 \ \forall i \in [k] \tag{17}$$

and round. We note that although LP (17) might have exponentially many constraints, it can be solved in polynomial time by using a minimum cut procedure to retrieve violating constraints.

In the online problem, the groups V_i arrive one at a time, and the goal is to find T' by irrevocably adding edges from E. The $O(\log^2 n \log k)$-competitive algorithm in [AAA+06] implicitly considers LP (17) in an online fashion by updating x in a non-decreasing manner and rounding x online.

In the learning-augmented problem, we are given a confidence parameter $\lambda \in [0,1]$ and a fractional advice $x' \in [0,1]^{|E|}$ indicating how likely each edge should be selected. Let OPT denote the weight of the minimum weighted subtree rooted at r that contains at least one vertex from each group V_i. We show the following.

Corollary 2.6. *For the learning-augmented online group Steiner tree problem on trees, there exists a polynomial time randomized online algorithm that generates $T' = (V', E', r)$ such that*

$$\sum_{e \in E'} c_e \leq \log n \log k \cdot \min \left\{ O\left(\frac{1}{1-\lambda}\right) c^T x' + O(\log n)\mathit{OPT}, O\left(\log \frac{n}{\lambda}\right)\mathit{OPT} \right\}$$

and $V' \cap V_i \neq \emptyset$ for each $i \in [k]$. If x' is feasible, then $\sum_{e \in E'} c_e \leq O\left(\frac{\log n \log k}{1-\lambda}\right) c^T x'$.

Proof. We use the following lemma due to [AAA+06].

Lemma 2.7. *Given an α-competitive feasible solution for LP (17), there exists an $O(\alpha \log n \log k)$-competitive randomized rounding scheme for the online group Steiner tree problem on trees.*

We use Theorem 2.4 to obtain an online solution x and observe that the sparsity parameter $s = O(|E|) = O(n^2)$. Let LP^* denote the optimal objective value of LP (17). We have that

$$\begin{aligned} c^T x &\le \min\left\{ O\left(\frac{1}{1-\lambda}\right) c^T x' + O(\log n^2)\mathsf{LP}^*, O\left(\log \frac{n^2}{\lambda}\right)\mathsf{LP}^* \right\} \\ &\le \min\left\{ O\left(\frac{1}{1-\lambda}\right) c^T x' + O(\log n)\mathsf{LP}^*, O\left(\log \frac{n}{\lambda}\right)\mathsf{LP}^* \right\} \end{aligned}$$

and $c^T x \le O\left(\frac{1}{1-\lambda}\right) c^T x'$ if x' is feasible for LP (17).

Combining the inequality above with Lemma 2.7, we have the desired guarantee for T'. □

3 PDLA Algorithms for Covering Semidefinite Programming

In this section, we prove Theorems 1.4 and 1.6. Namely, we present efficient, consistent, and robust PDLA algorithms for online covering SDPs. We recall that the online covering SDP problem (4) is the following.

$$\text{minimize } c^T x \text{ over } x \in \mathbb{R}^n_{\ge 0} \text{ subject to } \sum_{j=1}^n A_j x_j \succeq B^{(i)}.$$

Here, $A_j \in \mathbb{R}^{d\times d}_{\ge 0}$ is an SPSD matrix for each $j \in [n]$ and $c \in \mathbb{R}^n_{>0}$ denotes the positive coefficients of the linear cost function. We assume that in the beginning, $B^{(0)}$ is a zero matrix, and there are m arriving SPSD matrices $B^{(i)}$'s where m might be unknown. For each $i \in [m]$, it is required that $B^{(i)} \succeq B^{(i-1)}$. In round i, $B^{(i)}$ arrives and our goal is to approximately minimize the objective $c^T x$ by updating x in a non-decreasing manner such that x is feasible. An important idea is to simultaneously consider the dual packing SDP problem (5):

$$\text{maximize } B^{(i)} \otimes Y \text{ over } Y \succeq 0 \text{ subject to } A_j \otimes Y \le c_j \quad \forall j \in [n].$$

The approach closely follows the one for online covering LPs and the one in [EKN16]. In round i, either $\sum_{j=1}^n A_j x_j \succeq B^{(i)}$ and there is nothing to be done, or we use a subroutine to retrieve an implicit violating *linear* constraint and update x. More specifically, observe that $\sum_{j=1}^n A_j x_j \succeq B^{(i)}$ if and only if $v^T \left(\sum_{j=1}^n A_j x_j\right) v \ge v^T B^{(i)} v$ for all $v \in \mathbb{R}^n$, if the constraint in round i is violated, we can find an SPSD matrix $V = vv^T$ such that $v^T \left(\sum_{j=1}^n A_j x_j\right) v = \sum_{j=1}^n A_j x_j \otimes V < B^{(i)} \otimes V = v^T B^{(i)} v$, and we update x_j proportionally to $A_j \otimes V$. There are many possible options for V and without loss of generality, we can scale V by a factor of $\mathrm{trace}(V)$ such that $\mathrm{trace}(V) = 1$. One option is to use $V = vv^T$ where v is a unit vector corresponding to the smallest eigenvalue of $\sum_{j=1}^n A_j x_j - B^{(i)}$. Since $\sum_{j=1}^n A_j x_j - B^{(i)} \not\succeq 0$, its smallest eigenvalue λ_d is negative, so $(\sum_{j=1}^n A_j x_j - B^{(i)}) \otimes V = v(\sum_{j=1}^n A_j x_j - B^{(i)})v^T = \lambda_d < 0$ as required.

Again, we use the guess-and-double approach by increasing the primal and dual variables in each phase r. We estimate a lower bound $\alpha(r)$ for OPT in phase r. In phase 1, let $\alpha(1) \leftarrow \min_{j=1}^n \{\frac{c_j \,\mathrm{trace}(B^{(1)})}{\mathrm{trace}(A_j)}\}$ be a proper lower bound for OPT. Once the online objective exceeds $\alpha(r)$, we start the new phase $r+1$ from the current $B^{(i)}$ matrix (say phase $r+1$ starts with $i = i_{r+1}$, in particular, $i_1 = 1$), and double the estimated lower bound, i.e. $\alpha(r+1) \leftarrow 2\alpha(r)$. In the beginning of phase r, $x_j^{(r)} \leftarrow \min\{x'_j, \alpha(r)/(2nc_j)\}$.

We recall that x must be updated in a non-decreasing manner, so the algorithm maintains $\{x_j^{(r)}\}$, which denotes the value of each variable x_j in each phase r, and the value of each variable x_j is actually set to $\max_r\{x_j^{(r)}\}$.

In phase r and round i, we find V and increase Y continuously along V. If the advice is feasible, i.e., $\sum_{j=1}^{n} A_j x'_j \succeq B^{(i)}$, we increase each variable $x_j^{(r)}$ with growth rate

$$\frac{A_j \otimes V}{c_j}\left(x_j^{(r)} + \frac{\lambda B^{(i)} \otimes V}{\sum_{k=1}^{n} A_k \otimes V} + \frac{(1-\lambda) x'_j \mathbf{1}_{x_j^{(r)} < x'_j} B^{(i)} \otimes V}{\sum_{k=1}^{n} \mathbf{1}_{x_k^{(r)} < x'_k} A_k x'_k \otimes V}\right)$$

where $\mathbf{1}_{x_k^{(r)} < x'_k}$ is an indicator variable with value 1 if $x_k^{(r)} < x'_k$ and 0 otherwise. Alternatively if the advice is not feasible, we increase $x_j^{(r)}$ with growth rate

$$\frac{A_j \otimes V}{c_j}\left(x_j^{(r)} + \frac{B^{(i)} \otimes V}{\sum_{k=1}^{n} A_k \otimes V}\right).$$

We note that

$$\sum_{j=1}^{n} \frac{A_j \otimes V}{\sum_{k=1}^{n} A_k \otimes V} = 1 \text{ and } \sum_{j=1}^{n} \frac{A_j \otimes V\left(x'_j \mathbf{1}_{x_j^{(r)} < x'_j}\right)}{\sum_{k=1}^{n} \mathbf{1}_{x_k^{(r)} < x'_k} A_k x'_k \otimes V} = 1. \tag{18}$$

To implement with the proper growth rate, the dual variable Y initialized as a zero matrix is used as a proxy in each phase r. We increment $x^{(r)}$ until the implicit violating constraint i is satisfied *by a factor of 2*. This guarantees that at least one variable $x_j^{(r)}$ is doubled thus ensures that the algorithm only updates $x^{(r)}$ polynomially many times.

Similar to Algorithm 2, in phase r, each iteration ends in one of the following three cases: (1) the implicit violating linear constraint is satisfied by a factor of 2, (2) there exists a variable $x_j^{(r)}$ that reaches the advice value x'_j, or (3) the objective $c^T x^{(r)}$ reaches $\alpha(r)$. The coefficients D_j and B_j for fitting boundary conditions are defined based on the value of $x_j^{(r)}$ in the end of the previous iteration, stored as $\bar{x}_j$. The indicator value used for the growth rate, $\mathbf{1}_{x_j^{(r)} < x'_j}$, has the same value as $\mathbf{1}_{\bar{x}_j < x'_j}$ during an iteration.

The main algorithm that uses the phase scheme is presented in Algorithm 5. The continuous primal-dual approach in phase r and round i, used as a subroutine, is presented in Algorithm 6.

Algorithm 5 Phase Scheme for Algorithm 6

1: $r \leftarrow 1$, $\alpha(1) \leftarrow \min_{j\in[n]}\{c_j \operatorname{trace}(B^{(1)})/\operatorname{trace}(A_j)\}$, $Y \leftarrow 0$, $i_1 \leftarrow 1$.
2: **for** each $j \in [n]$ **do**
3: $\quad x_j^{(r)} \leftarrow \min\{x'_j, \alpha(r)/(2nc_j)\}$.
4: **for** arriving covering constraint $\sum_{j=1}^{n} A_j x_j \succeq B^{(i)}$ **do** ▷ $i = 1, 2, ..., m$ for an unknown m
5: $\quad$ Run Algorithm 6.
6: $\quad$ **if** $c^T x^{(r)} \geq \alpha(r)$ **then** ▷ start a new phase
7: $\quad\quad r \leftarrow r + 1$, $\alpha(r) \leftarrow 2\alpha(r-1)$, $Y \leftarrow 0$, $i_r \leftarrow i$.
8: $\quad\quad$ **for** each $j \in [n]$ **do**
9: $\quad\quad\quad x_j^{(r)} \leftarrow \min\{x'_j, \alpha(r)/(2nc_j)\}$.
10: $\quad\quad$ Go to line 5.
11: $\quad$ **for** each $j \in [n]$ **do**
12: $\quad\quad x_j \leftarrow \max_{\ell\in[r]}\{x_j^{(\ell)}\}$. ▷ this is the solution returned in each round i

The augmentation is in a continuous fashion, and it is not hard to implement it in a discrete way for any desired precision by binary search. In each iteration, we use binary search to find the proper δ, and $x^{(r)}$ is updated accordingly once. Therefore, to show that the algorithm is efficient, it suffices to bound the number of iterations. The performance of Algorithm 5 is stated in Theorem 3.1, the formal version of Theorem 1.4.

Algorithm 6 PDLA Online SDP Covering in Phase r and Round i

Input: $x^{(r)}$: current solution, $\alpha(r)$: estimate for OPT, $B^{(i)}$: current lower bound matrix, i_r: starting round of phase r, Y: dual variable matrix, λ: the confidence parameter, and x': the advice for x.
Output: Updated $x^{(r)}$ and Y.

1: $\bar{x} \leftarrow x^{(r)}$. ▷ $\bar{x}$ is the value of $x^{(r)}$ at the end of the previous iteration
2: **while** $\sum_{j=1}^{n} A_j x_j \not\succeq B^{(i)}$ **do**
3: Find an SPSD matrix V with $\mathrm{trace}(V) = 1$ such that $\sum_{j=1}^{n}(A_j x_j^{(r)}) \otimes V < B^{(i)} \otimes V$.
4: **for** each $j \in [n]$ **do**
5: **if** $\sum_{k=1}^{n} A_k x'_k \succeq B^{(i)}$ **then**
6: $D_j \leftarrow \frac{\lambda B^{(i)} \otimes V}{\sum_{k=1}^{n} A_k \otimes V} + \frac{(1-\lambda) x'_j \mathbf{1}_{\bar{x}_j < x'_j} B^{(i)} \otimes V}{\sum_{k=1}^{n} \mathbf{1}_{\bar{x}_k < x'_k} A_k x'_k \otimes V}$,
7: **else**
8: $D_j \leftarrow \frac{B^{(i)} \otimes V}{\sum_{k=1}^{n} A_k \otimes V}$.
9: $B_j \leftarrow \frac{\bar{x}_j + D_j}{\exp\left(\frac{A_j \otimes Y}{c_j}\right)}$.
10: **while** $\sum_{j=1}^{n} A_j x_j^{(r)} \otimes V < 2B^{(i)} \otimes V$ **do**
11: Set $\delta = 0$ and increase it continuously.
12: Increase Y by continuously adding $V\delta$ to Y.
13: Increase $x^{(r)}$ continuously by simultaneously setting

$$x_j^{(r)} \leftarrow B_j \exp\left(\frac{A_j \otimes Y}{c_j}\right) - D_j.$$

14: **if** any $x_j^{(r)} = x'_j$ **then**
15: Break and go to line 1.
16: **if** $c^T x^{(r)} \geq \alpha(r)$ **then**
17: Break and return.

Theorem 3.1. *For the learning-augmented online covering SDP problem, there exists an online algorithm that generates x such that*

$$c^T x \leq \min\left\{ O\left(\frac{1}{1-\lambda}\right) c^T x' + O\left(\log\left(\kappa n\right)\right) \mathit{OPT}, O\left(\log \frac{\kappa n}{\lambda}\right) \mathit{OPT} \right\}$$

and x is updated $O(n(\log \frac{c^T x}{\alpha(1)})(\log n + \log c^T x + \log \beta))$ times. If x' is feasible for SDP (4), then $c^T x \leq O(\frac{1}{1-\lambda}) c^T x'$. Here, $\beta = \max_{j\in[n]}\{a_j^{\max}/c_j\}$, $a_j^{\max} := \max_{i\in[m]}\{a_{ij} \mid a_{ij} > 0\}$, and

$$\kappa := \frac{\max_{j\in[n],i\in[m]}\{\lambda_{\max}(A_j), \lambda_{\max}(B^{(i)})\}}{\min_{j\in[n],i\in[m]}\{\lambda_{\min}(A_j), \lambda_{\min}(B^{(i)})\}}$$

where $\lambda_{\max}(M)$ and $\lambda_{\min}(M)$ are the positive maximum and minimum eigenvalues of a PSD matrix M, respectively.

Proof. Let $P(r) = c^T x^{(r)}$ and $D(r) = B^{(i)} \otimes Y$ be the objective value of the primal and the dual in phase r, respectively. We use Algorithm 5 and first show robustness, i.e., $c^T x \leq O\left(\log \frac{\kappa n}{\lambda}\right) \mathsf{OPT}$, then show consistency, i.e., $c^T x \leq O\left(\frac{1}{1-\lambda}\right) c^T x' + O(\log(\kappa n))\mathsf{OPT}$. Finally, we show that only $O(n(\log \frac{c^T x}{\alpha(1)})(\log n + \log c^T x + \log \beta))$ iterations are needed.

Robustness. To show that $c^T x \leq O\left(\log \frac{\kappa n}{\lambda}\right) \mathsf{OPT}$, we prove the following five claims:

(i) x is feasible for SDP (4).

(ii) For each *finished* phase r, $\alpha(r) \le 6D(r)$.

(iii) $Y/\Theta\left(\log \frac{\kappa n}{\lambda}\right)$ is feasible for SDP (5) in each phase.

(iv) The sum of the covering objective generated from phase 1 to r is at most $2\alpha(r)$.

(v) Let r' be the last phase, then the covering objective $c^T x \le 2\alpha(r')$.

The proofs of (iv) and (v) are the same as the ones in Theorem 2.1, hence omitted. Equipped with these claims and weak duality (6), we have that

$$c^T x \le \Theta(1)\alpha(r') \le \Theta(1)D(r') = O\left(\log \frac{\kappa n}{\lambda}\right) \mathsf{OPT}.$$

Proof of (i). We prove that $x^{(r)}$ is feasible in phase r by showing that the growing functions in Algorithm 6 line 13 increments $x^{(r)}$ in a continuous manner.

In the beginning of an iteration in round i, we have that

$$x_j^{(r)} = B_j \exp\left(\frac{A_j \otimes Y}{c_j}\right) - D_j = \frac{\bar{x}_j + D_j}{\exp\left(\frac{A_j \otimes Y}{c_j}\right)} \exp\left(\frac{A_j \otimes Y}{c_j}\right) - D_j = \bar{x}_j.$$

By following Algorithm 6, we have that $x^{(r)}$ is feasible since we terminate the while loop at line 2 when there does not exists an SPSD matrix V such that $\sum_{j=1}^n A_j x_j \otimes V < B^{(i)} \otimes V$, indicating that $\sum_{j=1}^n A_j x_j \succeq B^{(i)}$. Whenever we find such a V, it induces a linear constraint and we increment $x^{(r)}$ until it is satisfied by a factor of 2 as in Algorithm 6 line 10. As x is the coordinate-wise maximum of $\{x^{(r)}\}$ and the sum of matrices is still PSD, x must be feasible for SDP (4). □

Proof of (ii). In the beginning of phase r, $x_j^{(r)} = \min\{x'_j, \alpha(r)/(2nc_j)\}$, so $P(r)$ is initially at most $\alpha(r)/2$. The total increase of $P(r)$ is at least $\alpha(r)/2$ as $P(r) \ge \alpha(r)$ when phase r ends. Therefore, it suffices to show that

$$\frac{\partial P(r)}{\partial \delta} \le 3\frac{\partial D(r)}{\partial \delta}.$$

By considering the partial derivative of $P(r)$ with respect to δ, when $\sum_{j=1}^n A_j x'_j \succeq B^{(i)}$, we have that

$$\begin{aligned}
\frac{\partial P(r)}{\partial \delta} &= \sum_{j=1}^n c_j \frac{\partial x_j^{(r)}}{\partial \delta} \\
&= \sum_{j=1}^n c_j \frac{\partial}{\partial \delta}\left(B_j \exp\left(\frac{A_j \otimes (Y + V\delta)}{c_j}\right) - D_j\right) \\
&= \sum_{j=1}^n c_j \left(\frac{A_j \otimes V}{c_j}\right) B_j \exp\left(\frac{A_j \otimes (Y + V\delta)}{c_j}\right) \\
&= \sum_{j=1}^n (A_j \otimes V)\left(x_j^{(r)} + D_j\right) \\
&= \sum_{j=1}^n (A_j \otimes V)\left(x_j^{(r)} + \frac{\lambda B^{(i)} \otimes V}{\sum_{k=1}^n A_k \otimes V} + \frac{(1-\lambda)x'_j \mathbf{1}_{x_j^{(r)} < x'_j} B^{(i)} \otimes V}{\sum_{k=1}^n \mathbf{1}_{x_k^{(r)} < x'_k} A_k x'_k \otimes V}\right) \\
&\le (2 + \lambda + (1-\lambda))\, B^{(i)} \otimes V = 3\frac{\partial D(r)}{\partial \delta}
\end{aligned}$$

where the last inequality is due to the fact that $\sum_{j=1}^n A_j x_j \otimes V < 2B^{(i)} \otimes V$ and (18). The same result can be obtained when $\sum_{j=1}^n A_j x'_j \not\succeq B^{(i)}$ by regarding λ as 1. □

Proof of (iii). Y is the sum of matrices δV, where $\delta > 0$ and V is SPSD since $V = vv^T$. Hence, δV is SPSD. Y is the sum of SPSD matrices and is therefore itself SPSD.

It remains to show that $Y/\Theta(\log \frac{\kappa n}{\lambda})$ is feasible for SDP (5) after satisfying the last implicit linear constraint in phase r. We recall that within a phase, each iteration ends in one of the following two cases: (1) the implicit violating linear constraint is satisfied by a factor of 2, or (2) there exists a variable $x_j^{(r)}$ that reaches the advice value x'_j. Suppose there are t iterations in phase r. In iteration $p \le t$, let $x_j^{(r,p)}$, $B_j^{(p)}$, and $D_j^{(p)}$ be the value of $x_j^{(r)}$, B_j, and D_j in the end of iteration p, respectively. Additionally, let $Y^{(p)}$ be the value of Y in the end of iteration p. We have $x_j^{(r,p)} = B_j^{(p)} \exp\left(\frac{A_j \otimes Y^{(p)}}{c_j}\right) - D_j^{(p)}$. By using a similar argument in Theorem 2.1 (iii), we have that

$$\begin{aligned}
\frac{A_j \otimes Y^{(t)}}{c_j} &= \ln \frac{x_j^{(r,t)} + D_j^{(t)}}{B_j^{(t)}} = \ln \left(\frac{x_j^{(r,t)} + D_j^{(t)}}{x_j^{(r,t-1)} + D_j^{(t)}} \exp\left(\frac{A_j \otimes Y^{(t-1)}}{c_j}\right)\right) \\
&= \frac{A_j \otimes Y^{(t-1)}}{c_j} + \ln \frac{x_j^{(r,t)} + D_j^{(t)}}{x_j^{(r,t-1)} + D_j^{(t)}} = \dots \\
&= \ln \prod_{p=1}^{t} \frac{x_j^{(r,p)} + D_j^{(p)}}{x_j^{(r,p-1)} + D_j^{(p)}} \le \ln \frac{x_j^{(r,p)} + D_j^{min}}{x_j^{(r,0)} + D_j^{min}}
\end{aligned}$$

where $D_j^{min} := \min_{p \in [t]} \{D_j^{(p)}\}$.

Now we upper-bound $x_j^{(r,p)}$ and lower-bound D_j^{min} to obtain the robustness ratio. Notice that $x_j^{(r,p)} \le 2\kappa$ since whenever $\sum_{i=1}^n A_i x_i^{(r)} \otimes V < 2B^{(p)} \otimes V$, if we only consider $x_j^{(r,p)}$, the worst case is that the unit vector v such that $V = vv^T$, is in the same direction of the vector corresponding to the largest eigenvalue of $B^{(p)}$ and that of the smallest eigenvalue of A_i, and $x^{(r,p)}$ is incremented until the implicit linear constraint is satisfied by a factor of 2. We also have that

$$D_j^{min} \ge \frac{\lambda \min_{i \in [m]} \{\lambda_{\min}(B^{(i)})\}}{\sum_{k=1}^n \lambda_{\max}(A_i)} \ge \frac{\lambda}{\kappa n}.$$

With $x_j^{(r,0)} = \alpha(r)/(2nc_j) \ge 0$, we thus have that

$$\ln \frac{x_j^{(r,p)} + D_j^{min}}{x_j^{(r,0)} + D_j^{min}} \le \ln\left(1 + \frac{2\kappa}{D_j^{min}}\right) = O\left(\log \frac{\kappa^2 n}{\lambda}\right) = O\left(\log \frac{\kappa n}{\lambda} + \log \kappa\right) = O\left(\log \frac{\kappa n}{\lambda}\right).$$

□

Consistency. We then show that $c^T x \le O\left(\frac{1}{1-\lambda}\right) c^T x' + O\left(\log(\kappa n)\right) \mathsf{OPT}$. Suppose the algorithm is in phase r with constraint i arriving. If $\sum_{j=1}^n A_j x'_j \not\succeq B^{(i)}$, then the change of $c^T x$ simply follows (ii) and (iii) by regarding λ as 1, so $c^T x \le O\left(\log(\kappa n)\right) \mathsf{OPT}$. For the more interesting case when $\sum_{j=1}^n A_j x'_j \succeq B^{(i)}$, we decompose $\frac{\partial P(r)}{\partial y_i} = \frac{\partial P_c}{\partial y_i} + \frac{\partial P_u}{\partial y_i}$, where P_c is the component of the primal objective due to the advice and P_u is the component of the primal objective due to the online algorithm. We have that the rate of change is credited to ∂P_c if $x_j^{(r)} < x'_j$ and the rate of change is credited to ∂P_u otherwise, if $x_j^{(r)} \ge x'_j$. In particular,

$$\begin{aligned}
\frac{\partial P_c}{\partial \delta} &= \sum_{j \in [n]: x_j < x'_j} (A_j \otimes V) \left(x_j^{(r)} + \frac{\lambda B^{(i)} \otimes V}{\sum_{k=1}^n A_k \otimes V} + \frac{(1-\lambda) x'_j \mathbf{1}_{x_j^{(r)} < x'_j} B^{(i)} \otimes V}{\sum_{k=1}^n \mathbf{1}_{x_k^{(r)} < x'_k} A_k x'_k \otimes V} \right) \\
&\ge 0 + \left(\frac{\lambda}{\kappa n} + 1 - \lambda\right) B^{(i)} \otimes V,
\end{aligned}$$

$$\frac{\partial P_u}{\partial \delta} = \sum_{j\in[n]:x_j\geq x'_j} (A_j \otimes V)\left(x_j^{(r)} + \frac{\lambda B^{(i)}\otimes V}{\sum_{k=1}^n A_k\otimes V} + \frac{(1-\lambda)x'_j \mathbf{1}_{x_j^{(r)}<x'_j} B^{(i)}\otimes V}{\sum_{k=1}^n \mathbf{1}_{x_k^{(r)}<x'_k} A_k x'_k \otimes V}\right) \leq (2+\lambda)B^{(i)}\otimes V + 0.$$

Thus we have $\frac{\partial P_u}{\partial \delta} \leq \frac{(2+\lambda)B^{(i)}\otimes V}{(\lambda/(\kappa n)+1-\lambda)B^{(i)}\otimes V}\cdot \frac{\partial P_c}{\partial \delta}$, so that

$$\frac{\partial P(r)}{\partial \delta} \leq \left(1+\frac{2+\lambda}{1-\lambda}\right)\frac{\partial P_c}{\partial \delta} = O\left(\frac{1}{1-\lambda}\right)\frac{\partial P_c}{\partial \delta}.$$

We note that if x' is feasible, then $\sum_{j=1}^n A_j x'_j \succeq B^{(i)}$, so $c^T x \leq O\left(\frac{1}{1-\lambda}\right)c^T x'$.

Bounding the number of iterations. The analysis simply follows the one for Theorem 2.1. The main difference is that (1) instead of violating linear covering constraints, we use implicit violating linear constraints, and (2) we bound the number of iterations instead of implicit violating constraints, where the number of iterations is $O(n)$ more than the number of implicit violating constraints since each variable can reach the advice value once. □

3.1 Adding Box Constraints

We recall that the covering SDP problem (8) with box constraints is the following.

$$\text{minimize } c^T x \text{ over } x\in[0,1]^n \text{ subject to } \sum_{j=1}^n A_j x_j \succeq B^{(i)}.$$

Our PDLA algorithm simultaneously considers the dual packing SDP problem:

$$\text{maximize } B^{(i)}\otimes Y - \mathbf{1}^T z \text{ over } Y\succeq 0 \text{ and } z\in\mathbb{R}^n_{\geq 0} \text{ subject to } A_j\otimes Y - z_j \leq c_j \quad \forall j\in[n]. \tag{19}$$

We recall that we assume that the advice $x'\in[0,1]^n$.

We use the guess-and-double approach similar to Algorithms 3 and 5. We maintain the set T which denotes the subscript indices of the x variables that are tight. In phase r, upon the arrival of an implicit violating linear constraint, we increment $x_j^{(r)}$ in terms of an exponential function of δ and z_j subject to $x_j^{(r)}\leq 1$. Once $x_j^{(r)}=1$, we add j to the tight set T and stop incrementing $x_j^{(r)}$, but we still increment Y continuously and z_j with rate $A_j\otimes V\delta$ in order to maintain dual feasibility. In the beginning of each phase, z is a zero vector and T is an empty set. $z_j=0$ whenever $j\in[n]\setminus T$. Subject to $x_j^{(r)}\leq 1$, $x_j^{(r)}$ is increasing until the cost *outside of the tight set* exceeds the *remaining capacity* by a factor of 2. More specifically, when we have an implicit violating linear constraint induced by V, we have that $\sum_{j\in[n]\setminus T} A_j x_j^{(r)}\otimes V < B^{(i)}\otimes V - \sum_{j\in T} A_j\otimes V$, and we increment until $\sum_{j\in[n]\setminus T} A_j x_j^{(r)}\otimes V \geq 2(B^{(i)}\otimes V - \sum_{j\in T} A_j\otimes V)$.

If the advice is feasible, i.e., $\sum_{j=1}^n A_j x'_j \succeq B^{(i)}$, we increase each variable $x_j^{(r)}$ with growth rate

$$\frac{A_j\otimes V}{c_j}\left(x_j^{(r)} + \left(\frac{\lambda}{\sum_{k\in[n]\setminus T} A_k\otimes V} + \frac{(1-\lambda)x'_j\mathbf{1}_{x_j^{(r)}<x'_j}}{\sum_{k\in[n]\setminus T}\mathbf{1}_{x_k^{(r)}<x'_k}A_k x'_k\otimes V}\right)\left(B^{(i)}\otimes V - \sum_{k\in T} A_k\otimes V\right)\right).$$

Alternatively if the advice is not feasible, we increase $x_j^{(r)}$ with growth rate

$$\frac{A_j\otimes V}{c_j}\left(x_j^{(r)} + \frac{B^{(i)}\otimes V - \sum_{k\in T} A_k\otimes V}{\sum_{k\in[n]\setminus T} A_k\otimes V}\right).$$

Similar to (18), we have that

$$\sum_{j\in[n]\setminus T} \frac{A_j \otimes V}{\sum_{k\in[n]\setminus T} A_k \otimes V} = 1 \text{ and } \sum_{j\in[n]\setminus T} \frac{x'_j \mathbf{1}_{x_j^{(r)}<x'_j} A_j \otimes V}{\sum_{k\in[n]\setminus T} \mathbf{1}_{x_k^{(r)}<x'_k} A_k x'_k \otimes V} = 1. \tag{20}$$

Each iteration now ends in one of the following four cases: (1) the implicit linear constraint induces a cost outside of the tight set that exceeds the remaining capacity by a factor of 2, (2) there exists a variable $x_j^{(r)}$ that reaches the advice value x'_j, (3) there exists a variable $x_j^{(r)}$ that reaches 1, or (4) the objective $c^T x^{(r)}$ reaches $\alpha(r)$. The coefficients D_j and B_j for fitting boundary conditions are defined based on the value of $x_j^{(r)}$ in the end of the previous iteration, stored as $\bar{x}_j$. Again, the indicator value used for the growth rate, $\mathbf{1}_{x_j^{(r)}<x'_j}$, has the same value as $\mathbf{1}_{\bar{x}_j<x'_j}$ during an iteration.

The main algorithm that uses the phase scheme is presented in Algorithm 7. The continuous primal-dual approach in phase r and round i, used as a subroutine, is presented in Algorithm 8. The performance of Algorithm 7 is stated in Theorem 3.2, the formal version of Theorem 1.6. We note that in line 3 of Algorithm 8, given the round i and the tight set T, we find V such that $\frac{\sum_{j\in[n]\setminus T} A_j\otimes V}{B^{(i)}\otimes V - \sum_{j\in T} A_j\otimes V}$ is minimized.[3]

Algorithm 7 Phase Scheme for Algorithm 8

1: $r \leftarrow 1$, $\alpha(1) \leftarrow \min_{j\in[n]}\{c_j \operatorname{trace}(B^{(1)})/\operatorname{trace}(A_j)\}$, $Y \leftarrow 0$, $T \leftarrow \emptyset$, $i_1 \leftarrow 1$.
2: **for** each $j \in [n]$ **do**
3: $\quad x_j^{(r)} \leftarrow \min\{x'_j, \alpha(r)/(2nc_j)\}$, $z_j \leftarrow 0$.
4: **for** arriving covering constraint $\sum_{j=1}^n A_j x_j \succeq B^{(i)}$ **do** ▷ $i = 1, 2, ..., m$ for an unknown m
5: $\quad$ Run Algorithm 8.
6: $\quad$ **if** $c^T x^{(r)} \geq \alpha(r)$ **then** ▷ start a new phase
7: $\quad\quad r \leftarrow r+1$, $\alpha(r) \leftarrow 2\alpha(r)$, $Y \leftarrow 0$, $T \leftarrow \emptyset$, $i_r \leftarrow i$.
8: $\quad\quad$ **for** each $j \in [n]$ **do**
9: $\quad\quad\quad x_j^{(r)} \leftarrow \min\{x'_j, \alpha(r)/(2nc_j)\}$, $z_j \leftarrow 0$.
10: $\quad\quad$ Go to line 5.
11: $\quad$ **for** each $j \in [n]$ **do**
12: $\quad\quad x_j \leftarrow \max_{\ell\in[r]}\{x_j^{(\ell)}\}$. ▷ this is the solution returned in each round i

Theorem 3.2. *For the learning-augmented online covering SDP problem with box constraints, there exists an online algorithm that generates x such that*

$$c^T x \leq \min\{O(\frac{1}{1-\lambda})c^T x' + O(\log s)\mathit{OPT}, O(\log \frac{s}{\lambda})\mathit{OPT}\}$$

and x is updated $O(n \log s \log \frac{c^T x}{\alpha(1)})$ times. If x' is feasible for SDP (8), then $c^T x \leq O(\frac{1}{1-\lambda})c^T x'$. Here,

$$s := \max_{i\in[m], T\subseteq[n]\, V: B^{(i)}\otimes V - \sum_{j\in T} A_j\otimes V > 0} \min \left\{ \frac{\sum_{j\in[n]\setminus T} A_j \otimes V}{B^{(i)} \otimes V - \sum_{j\in T} A_j \otimes V} \right\}.$$

Proof. Let $P(r) = c^T x^{(r)}$ and $D(r) = B^{(i)} \otimes Y - \mathbf{1}^T z$ be the objective value of the primal and the dual in phase r, respectively. We use Algorithm 7 and first show robustness, i.e., $c^T x \leq O\left(\log \frac{s}{\lambda}\right) \mathsf{OPT}$, then

[3] Following [EKN16], for fixed i and T, V can be calculated by estimating η to any desired precision via solving the SDP:

$$\min_V \left\{ \left(\sum_{j\in[n]\setminus T} A_j - \eta \left(B^{(i)} - \sum_{j\in T} A_j \right) \right) \otimes V \mid V \succeq 0, \left(B^{(i)} - \sum_{j\in T} A_j \right) \otimes V \geq 0 \right\}.$$

Algorithm 8 PDLA Online SDP Covering with Box Constraints in Phase r and Round i

Input: $x^{(r)}$: current solution, $\alpha(r)$: estimate for OPT, $B^{(i)}$: current lower bound matrix, i_r: starting round of phase r, Y: dual variable matrix, z: dual variable vector, T: tight variable set, λ: the confidence parameter, and x': the advice for x.
Output: Updated $x^{(r)}$, Y, and z.

1: $\bar{x} \leftarrow x^{(r)}$. ▷ $\bar{x}$ is the value of $x^{(r)}$ at the end of the previous iteration
2: **while** $\sum_{j=1}^{n} A_j x_j \not\succeq B^{(i)}$ **do**
3: Find an SPSD matrix V with $\mathrm{trace}(V) = 1$ such that $\sum_{j=1}^{n}(A_j x_j^{(r)}) \otimes V < B^{(i)} \otimes V$.
4: **for** each $j \in [n]$ **do**
5: **if** $\sum_{k=1}^{n} A_k x'_k \succeq B^{(i)}$ **then**
6: $D_j \leftarrow \left(\frac{\lambda}{\sum_{k\in[n]\setminus T} A_k \otimes V} + \frac{(1-\lambda)x'_j \mathbf{1}_{\bar{x}_j < x'_j}}{\sum_{k\in[n]\setminus T} \mathbf{1}_{\bar{x}_k < x'_k} A_k x'_k \otimes V} \right) \left(B^{(i)} \otimes V - \sum_{k\in T} A_k \otimes V \right)$,
7: **else**
8: $D_j \leftarrow \frac{B^{(i)} \otimes V - \sum_{k\in T} A_k \otimes V}{\sum_{k\in[n]\setminus T} A_k \otimes V}$.
9: $B_j \leftarrow \frac{\bar{x}_j + D_j}{\exp((A_j \otimes Y - z_j)/c_j)}$.
10: **while** $T \neq [n]$ and $\sum_{j\in[n]\setminus T} A_j x_j^{(r)} \otimes V < 2(B^{(i)} \otimes V - \sum_{j\in T} A_j \otimes V)$ **do**
11: Set $\delta = 0$ and increase it continuously.
12: Increase Y by continuously adding $V\delta$ to Y.
13: **for** each $j \in T$ **do**
14: Increase z_j with rate $A_j \otimes V\delta$.
15: Update $x^{(r)}$ continuously by simultaneously setting

$$x_j^{(r)} \leftarrow \min\left\{1, B_j \exp\left(\frac{A_j \otimes Y - z_j}{c_j}\right) - D_j\right\}.$$

16: **if** any $x_j^{(r)} = x'_j < 1$ **then**
17: Break and go to line 1.
18: **if** any $x_j^{(r)} = 1$ for $j \notin T$ **then**
19: Add j to T and go to line 1.
20: **if** $c^T x^{(r)} \geq \alpha(r)$ **then**
21: Break and return.
22: **if** $T = [n]$ and $\sum_{j=1}^{n} A_j x_j \not\succeq B^{(i)}$ **then**
23: **return** no feasible solution.

show consistency, i.e., $c^T x \leq O\left(\frac{1}{1-\lambda}\right) c^T x' + O(\log s)\mathsf{OPT}$. Finally, we show that only $O(n \log s \log \frac{c^T x}{\alpha(1)})$ iterations are needed.

Robustness. To show that $c^T x \leq O\left(\log \frac{s}{\lambda}\right) \mathsf{OPT}$, we prove the following five claims:

(i) If line 23 in Algorithm 8 is not encountered, then x is feasible for SDP (8).

(ii) For each *finished* phase r, $\alpha(r) \leq 6D(r)$.

(iii) $(y, z)/\Theta\left(\log \frac{s}{\lambda}\right)$ is feasible for SDP (19) in each phase.

(iv) The sum of the covering objective generated from phase 1 to r is at most $2\alpha(r)$.

(v) Let r' be the last phase, then the covering objective $c^T x \leq 2\alpha(r')$.

The proofs of (iv) and (v) are the same as the ones in Theorem 3.1, hence omitted. Equipped with these claims and weak duality (6), we have that

$$c^T x \le \Theta(1)\alpha(r') \le \Theta(1)D(r') = O\left(\log \frac{s}{\lambda}\right)\mathsf{OPT}.$$

Proof of (i). We prove that $x^{(r)}$ is feasible in phase r by showing that the growing functions in line 16 increments $x^{(r)}$ in a continuous manner. In the beginning of an iteration in round i, we have that

$$x_j^{(r)} = B_j \exp\left(\frac{A_j \otimes Y}{c_j}\right) - D_j = \frac{\bar{x}_j + D_j}{\exp\left(\frac{A_j \otimes Y}{c_j}\right)} \exp\left(\frac{A_j \otimes Y}{c_j}\right) - D_j = \bar{x}_j.$$

By following Algorithm 8, we have that $x^{(r)}$ is feasible since we terminate the while loop at line 2 when there does not exists an SPSD matrix V such that $\sum_{j=1}^n A_j x_j \otimes V < B^{(i)} \otimes V$, indicating that $\sum_{j=1}^n A_j x_j \succeq B^{(i)}$. Whenever we find such a V, it induces a linear constraint and we increment $x^{(r)}$ until the cost outside of the tight set exceeds the remaining capacity by a factor of 2 in line 10. As x is the coordinate-wise maximum of $\{x^{(r)}\}$, capped at 1, and the sum of SPSD matrices is still SPSD, x must be feasible for SDP (8). □

Proof of (ii). In the beginning of phase r, $x_j^{(r)} = \min\{x'_j, \alpha(r)/(2nc_j)\}$, so $P(r)$ is initially at most $\alpha(r)/2$. The total increase of $P(r)$ is at least $\alpha(r)/2$ as $P(r) \ge \alpha(r)$ when phase r ends. Therefore, it suffices to show that

$$\frac{\partial P(r)}{\partial \delta} \le 3\frac{\partial D(r)}{\partial \delta}.$$

Recall that T is the set of the tight x indices and the z_j variables with in $j \in T$ are increasing with rate $A_j \otimes V$, we have that

$$\frac{\partial D(r)}{\partial \delta} = B^{(i)} \otimes V - \sum_{j \in T} A_j \otimes V.$$

When Y is increasing, this partial derivative is always non-negative due to the condition in Algorithm 8 line 10. Namely, at the point when $x_j^{(r)}$ is tight, we add j to T, and this may make the partial derivative change from non-negative to negative. In this case, we have that $2(B^{(i)} \otimes V - \sum_{j\in T} A_j \otimes V) < 0$ so the condition automatically fails. From here, when $\sum_{j=1}^n A_j x'_j \succeq B^{(i)}$, we have that

$$\begin{aligned}
\frac{\partial P(r)}{\partial \delta} &= \sum_{j\in[n]\setminus T} c_j \frac{\partial x_j^{(r)}}{\partial \delta} = \sum_{j\in[n]\setminus T} c_j \frac{\partial}{\partial \delta}\left(B_j \exp\left(\frac{A_j \otimes (Y + V\delta) - z_j}{c_j}\right) - D_j\right) \\
&= \sum_{j\in[n]\setminus T} c_j \left(\frac{A_j \otimes V}{c_j}\right) B_j \exp\left(\frac{A_j \otimes (Y + V\delta) - z_j}{c_j}\right) = \sum_{j\in[n]\setminus T} (A_j \otimes V)(x_j^{(r)} + D_j) \\
&= \sum_{j\in[n]\setminus T} (A_j \otimes V)(x_j^{(r)} + \frac{\lambda(B^{(i)} \otimes V - \sum_{k\in T} A_k \otimes V)}{\sum_{k\in[n]\setminus T} A_k \otimes V} + \frac{(1-\lambda)x'_j \mathbf{1}_{x_j^{(\ell)} < x'_j}(B^{(i)} \otimes V - \sum_{k\in T} A_k \otimes V)}{\sum_{k\in[n]\setminus T} \mathbf{1}_{x_k^{(\ell)} < x'_k} A_k x'_k \otimes V}) \\
&\le (2 + \lambda + (1-\lambda))\,(B^{(i)} \otimes V - \sum_{j\in T} A_j \otimes V) = 3\frac{\partial D(r)}{\partial \delta}
\end{aligned}$$

where the last inequality is due to the fact that $\sum_{j\in[n]\setminus T} A_j x_j \otimes V < 2(B^{(i)} \otimes V - \sum_{j\in T} A_j \otimes V)$ and (20). The same result can be obtained when $\sum_{j=1}^n A_j x'_j \not\succeq B^{(i)}$ by regarding λ as 1. □

Proof of (iii). The constraint $Y \succeq 0$ is satisfied by the same statement in Theorem 3.1. We show that $y/\Theta(\log \frac{\kappa n}{\lambda})$ is feasible for SDP (19) after satisfying the last implicit linear constraint in phase r. We recall that within a phase r, each iteration ends in one of the following three cases: (1) the implicit linear constraint

induces a cost outside of the tight set that exceeds the remaining capacity by a factor of 2, (2) there exists a variable $x_j^{(r)}$ that reaches the advice value x'_j, or (3) there exists a variable $x_j^{(r)}$ that reaches 1. Suppose there are t iterations in phase r. In iteration $p \le t$, let $x_j^{(r,p)}$, $B_j^{(p)}$, and $D_j^{(p)}$ be the value of $x_j^{(r)}$, B_j, and D_j in the end of iteration p, respectively. Additionally, let $Y^{(p)}$ and $z_j^{(p)}$ be the value of Y and z_j in the end of iteration p, respectively. We have $x_j^{(r,p)} = B_j^{(p)} \exp\left(\frac{A_j \otimes Y^{(p)} - z_j^{(p)}}{c_j}\right) - D_j^{(p)}$. By using a similar argument in Theorem 3.1 (iii), we have that

$$\begin{aligned}
\frac{A_j \otimes Y^{(t)} - z_j^{(t)}}{c_j} &= \ln \frac{x_j^{(r,t)} + D_j^{(t)}}{B_j^{(t)}} = \ln\left(\frac{x_j^{(r,t)} + D_j^{(t)}}{x_j^{(r,t-1)} + D_j^{(t)}} \exp\left(\frac{A_j \otimes Y^{(t-1)} - z_j^{(t-1)}}{c_j}\right)\right) \\
&= \frac{A_j \otimes Y^{(t-1)} - z_j^{(t-1)}}{c_j} + \ln \frac{x_j^{(r,t)} + D_j^{(t)}}{x_j^{(r,t-1)} + D_j^{(t)}} = \dots \\
&= \ln \prod_{p=1}^{p} \frac{x_j^{(r,p)} + D_j^{(p)}}{x_j^{(r,p-1)} + D_j^{(p)}} \le \ln \frac{x_j^{(r,p)} + D_j^{min}}{x_j^{(r,0)} + D_j^{min}}
\end{aligned}$$

where $D_j^{min} := \min_{p\in[t]}\{D_j^{(p)}\}$.

Now we upper-bound $x_j^{(r,p)}$ and lower-bound D_j^{min} to obtain the robustness ratio. Clearly, $x_j^{(r,p)} \le 1$. We also have that

$$\begin{aligned}
D_j^{min} &\ge \frac{\lambda\left(B^{(i)} \otimes V - \sum_{k\in T} A_k \otimes V\right)}{\sum_{k\in[n]\setminus T} A_k \otimes V} \\
&\ge \min_{i\in[m], T\subseteq[n]} \max_{V: B^{(i)}\otimes V - \sum_{k\in T} A_k \otimes V > 0} \left\{\frac{\lambda\left(B^{(i)} \otimes V - \sum_{k\in T} A_k \otimes V\right)}{\sum_{k\in[n]\setminus T} A_k \otimes V}\right\} \ge \frac{\lambda}{s}
\end{aligned}$$

where the second inequality holds by considering all possible i and T and using our choice of V.

With $x_j^{(r,0)} \in [0,1]$, we thus have that

$$\ln \frac{x_j^{(r,p)} + D_j^{min}}{x_j^{(r,0)} + D_j^{min}} \le \ln\left(1 + \frac{1}{D_j^{min}}\right) = O\left(\log \frac{s}{\lambda}\right).$$

□

Consistency. We then show that $c^T x \le O\left(\frac{1}{1-\lambda}\right) c^T x' + O(\log s)\mathsf{OPT}$. Suppose the algorithm is in phase r with constraint i arriving. If $\sum_{j=1}^n A_j x'_j \not\succeq B^{(i)}$, then the change of $c^T x$ simply follows (ii) and (iii) by regarding λ as 1, so $c^T x \le O(\log s)\mathsf{OPT}$. For the more interesting case when $\sum_{j=1}^n A_j x'_j \succeq B^{(i)}$, we decompose $\frac{\partial P(r)}{\partial y_i} = \frac{\partial P_c}{\partial y_i} + \frac{\partial P_u}{\partial y_i}$, where P_c is the component of the primal objective due to the advice and P_u is the component of the primal objective due to the online algorithm. We have that the rate of change is credited to ∂P_c if $x_j^{(r)} < x'_j$ and the rate of change is credited to ∂P_u otherwise, if $x_j^{(r)} \ge x'_j$. In particular,

$$\begin{aligned}
\frac{\partial P_c}{\partial \delta} &= \sum_{j\in[n]: x_j < x'_j} (A_j \otimes V)\left(x_j^{(r)} + \frac{\lambda\left(B^{(i)} \otimes V - \sum_{k\in T} A_k \otimes V\right)}{\sum_{k\in[n]\setminus T} A_k \otimes V} + \frac{(1-\lambda) x'_j \mathbf{1}_{x_j^{(r)} < x'_j}\left(B^{(i)} \otimes V - \sum_{k\in T} A_k \otimes V\right)}{\sum_{k\in[n]\setminus T} \mathbf{1}_{x_k^{(r)} < x'_k} A_k x'_k \otimes V}\right) \\
&\ge 0 + \left(\frac{\lambda}{s} + 1 - \lambda\right)\left(B^{(i)} \otimes V - \sum_{k\in T} A_k \otimes V\right),
\end{aligned}$$

$$\frac{\partial P_u}{\partial \delta} = \sum_{j\in[n]:x_j\geq x'_j} (A_j \otimes V)\left(x_j^{(r)} + \frac{\lambda\left(B^{(i)}\otimes V - \sum_{k\in T} A_k\otimes V\right)}{\sum_{k\in[n]\setminus T} A_k\otimes V} + \frac{(1-\lambda)x'_j \mathbf{1}_{x_j^{(r)}<x'_j}\left(B^{(i)}\otimes V - \sum_{k\in T} A_k\otimes V\right)}{\sum_{k\in[n]\setminus T}\mathbf{1}_{x_k^{(r)}<x'_k} A_k x'_k\otimes V}\right)$$
$$\leq (2+\lambda)\left(B^{(i)}\otimes V - \sum_{k\in T} A_k\otimes V\right) + 0.$$

Thus we have $\frac{\partial P_u}{\partial \delta} \leq \frac{(2+\lambda)\left(B^{(i)}\otimes V - \sum_{k\in T} A_k\otimes V\right)}{(\lambda/s+1-\lambda)\left(B^{(i)}\otimes V - \sum_{k\in T} A_k\otimes V\right)} \cdot \frac{\partial P_c}{\partial \delta}$, so that

$$\frac{\partial P(r)}{\partial \delta} \leq \left(1+\frac{2+\lambda}{1-\lambda}\right)\frac{\partial P_c}{\partial \delta} = O\left(\frac{1}{1-\lambda}\right)\frac{\partial P_c}{\partial \delta}.$$

We note that if x' is feasible, then $\sum_{j=1}^n A_j x'_j \succeq B^{(i)}$, so $c^T x \leq O\left(\frac{1}{1-\lambda}\right) c^T x'$.

Bounding the number of iterations. The analysis follows similarly to the one for Theorem 2.4. We first have that there are $O(\log(c^T x/\alpha(1)))$ phases.

We recall that within phase r, each iteration ends in one of the following three cases: (1) the implicit linear constraint induces a cost outside of the tight set that exceeds the remaining capacity by a factor of 2, (2) there exists a variable $x_j^{(r)}$ that reaches the advice value x'_j, (3) there exists a variable $x_j^{(r)}$ that reaches 1. If an iteration ends in the first case, before $x^{(r)}$ is updated, we have an implicit violating linear constraint induced by V such that $\sum_{j\in[n]\setminus T} A_j x_j^{(r)}\otimes V < B^{(i)}\otimes V - \sum_{j\in T} A_j\otimes V$. We increment $x_j^{(r)}$'s where $j\in[n]\setminus T$ until $\sum_{j\in[n]\setminus T} A_j x_j^{(r)}\otimes V \geq 2\left(B^{(i)}\otimes V - \sum_{j\in T} A_j\otimes V\right)$. One of the following two cases must hold after updating $x^{(r)}$: (1) there exist a *large* variable $x_j^{(r)} \geq 1/(2s)$ that is updated to at least $3x_j^{(r)}/2$, or (2) there exists a *small* variable $x_j^{(r)} < 1/(2s)$ that becomes large, i.e., $x_j^{(r)}$ is updated to at least $1/(2s)$. Let L and S be the set of large and small variable subscript labels in $[n]\setminus T$ before the violating constraint i arrives, respectively, and $\hat{x}_j^{(r)}$ be the value of $x_j^{(r)}$ after the update. If none of these two cases holds, then

$$\begin{aligned}\sum_{j\in[n]\setminus T} A_j\hat{x}_j^{(r)}\otimes V &< \frac{3}{2}\sum_{j\in L} A_j x_j^{(r)}\otimes V + \frac{1}{2s}\sum_{j\in S} A_j\otimes V\\ &< \frac{3}{2}\left(B^{(i)}\otimes V - \sum_{j\in T} A_j\otimes V\right) + \frac{\left(B^{(i)}\otimes V - \sum_{j\in T} A_j\otimes V\right)}{2\sum_{j\in[n]\setminus T} A_j\otimes V}\sum_{j\in S} A_j\otimes V\\ &\leq 2\left(B^{(i)}\otimes V - \sum_{j\in T} A_j\otimes V\right)\end{aligned}$$

where the second inequality is by the fact that implicit linear constraint induced by V is violated before the update and the definition of the sparsity s, while the last inequality is by $S\subseteq[n]\setminus T$ and $A_j\otimes V\geq 0$ since A_j is SPSD. This implies that the cost outside of the tight set does not exceed the remaining capacity by a factor of 2 after the update, a contradiction.

Suppose $x_j^{(r)}$ has been updated t times by a multiplicative factor of 3/2 since it was large in phase r, then $t = O(\log s)$ since $\frac{1}{2s}(\frac{3}{2})^t \leq 1$.

There are n variables. In each phase, each variable can be updated from small to large once, updated at most t times by a factor of 3/2 since it was large, capped at 1 once, and reach the advice value once. Hence, Algorithm 7 has $O(n\log s\log\frac{c^T x}{\alpha(1)})$ iterations. □

4 Empirical Evaluations

We demonstrate the applicability of our algorithmic framework on a synthetic dataset as well as a real world case study on internet router graphs. Our focus will be on evaluating our online covering algorithms, Algorithms 1 and 2, with possibly fractional hints and fractional entries. We focus on this setting since it's the simplest of our algorithms and already captures our overall algorithm framework. Note that prior work in [BMS20] has already demonstrated the empirical advantage of learning-based methods for online covering with integral hints and constraints, albeit on synthetic datasets.

Datasets. Our synthetic dataset is constructed as follows. The constraint matrix A represents a $n \times n$ matrix where each entry is uniformly in $\{0, 1\}$. We set $n = 500$. The objective function c is a scaled vector with entries uniform in $[0, 1]$. The vector we are covering is the all ones vector, i.e., we are solving $Ax \geq \mathbf{1}$, and the rows of A arrive online. Our graph dataset is constructed as follows. We have a sequence of nine (unweighted) graphs which represents an internet router network sampled across time [LK14, LKF05][4]. The graphs have approximately $n \sim 10^4$ nodes and $m \sim 2.2 \cdot 10^4$ edges. We note that the nodes of the graphs are labeled and the labeling is consistent throughout the different time stamps. Each graph defines an instance of the set cover problem derived from vertex cover as follows. The edges of the graphs are labeled and the n vertex neighborhoods define n sets. The edges of the graph (the universe elements) appear online and we must cover these edges by choosing vertices (the sets) which are incident on them. The objective function c will be the same as the synthetic case so our problem represents an instance of weighted vertex cover.

Predictions. We instantiate predictions in a variety of ways, drawing inspiration from many prior works [BMS20, DIL+21, CSVZ22]. First we describe our predictions for the synthetic datasets. We consider two types of predictions. In one case, we first find the optimal offline solution x by solving the full linear program. We then noisy corrupt the entries of x by setting the entries to be 0 independently with some probability p. This is the same type of predictions in [BMS20] which they refer to as the 'replacement rate' strategy.

The second type of predictions are informed by the following motivating setting of learning-augmented algorithms: we are solving many different, but related, problem instances. To mimic this situation, we have matrices $A_0, A_1, \ldots$ where each index represents a new problem instance. A_0 is our synthetic matrix and A_{i+1} updates A_i by flipping n entries at random. We fix c to be the same throughout. The predictions for all instances $i \geq 1$ are given by the optimal offline solution generated from the first instance A_0. This "batch" experimental design naturally models the scenario where the current problem instance is similar to past instances and so one can hope to utilize past learned information to improve current algorithmic performance. A similar style of predictions, although not in an online context, has been employed in [CEI+22, EFS+22, DIL+21, CSVZ22]. Furthermore, this mimics the setting for proving PAC learning bounds.

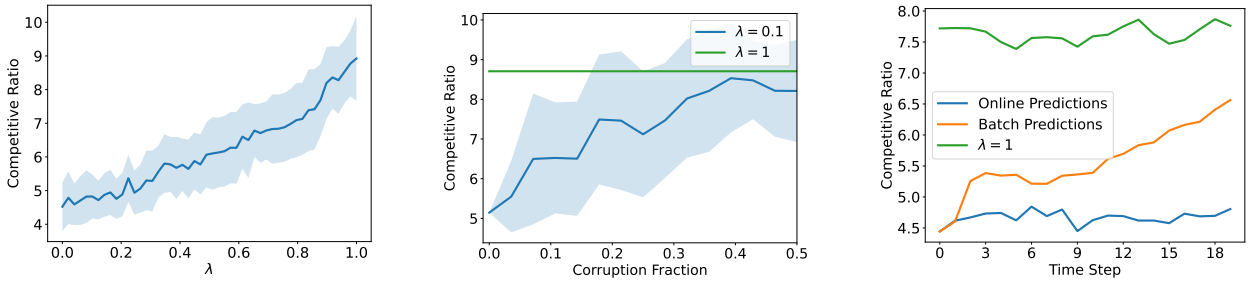
For our graph dataset, we first solve the set cover instance on the first graph in the family using an offline linear program solver. We then use the solution from the first graph as the hint for all subsequent graphs. We also noisy alter this hint for one of our experiments using the replacement rate strategy. Note that the set of vertices might vary across graphs since new vertices can appear in the network while older vertices may be removed. In this case, we set the corresponding entry in the hint vector to be 0.

The type of dataset and predictions used will always be stated explicitly in our experimental results below.

Results. Our results are shown in Figures 2 and 3. Figure 2 shows the results on the synthetic dataset while Figure 3 refers to our graph dataset. Description of each figure follows.

In Figure 2a, we consider a single online instance of the synthetic dataset. Our prediction is the offline optimal solution. The figure shows a smooth trade-off in the competitive ratio as the parameter λ ranges from 0 (full trust in the predictions) to 1 (standard online algorithm with no hints), as predicted by our theoretical bounds. Since the instance is random, we plot the average of 20 trials for each setting of λ and

[4] Graphs can be accessed in https://snap.stanford.edu/data/Oregon-1.html



(a) (b) (c)

Figure 2: Figures for our synthetic dataset.

also shade in one standard deviation. The plot validates the consistency of our algorithms as the competitive ratio is a factor of two lower with accurate predictions.

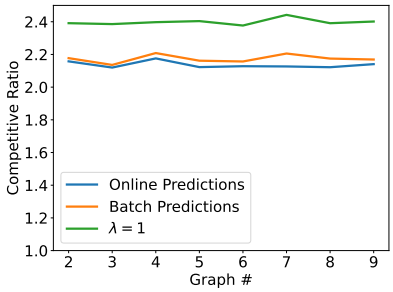
In Figure 2b, we again consider a single synthetic instance. This time, we consider the "replacement rate" strategy and randomly zero out the entries of the prediction (which is again the offline optimum) independently. The expected fraction of entries in the hint vector being set to zero is denoted as the corruption rate and is shown in the x-axis. We see that for a fixed setting of λ, such as $\lambda = 0.1$, our algorithm performs much better than not utilizing any hints when the corruption factor is low. This intuitively makes sense and is exactly what Figure 2a shows. However, as we increase the corruption factor, the performance of the algorithm degrades. Crucially, the performance *does not* degrade arbitrarily worse compared to $\lambda = 1$ (no hints) performance, which empirically validates the robustness of our algorithm. Indeed, our algorithm with hints is able to outperform the baseline even up to a high corruption factor. Lastly, if we utilize hints in a naive way where we only scale the hint variables to satisfy the constraints, then the competitive ratio is at least four orders of magnitude larger than the values in Figure 2b. Thus, post-processing the hints, such as what our algorithm does, is crucial.

In Figure 2c, we consider a "batch" experimental design for our synthetic dataset with 20 time steps. The green curve shows the competitive ratio without using any hints, i.e., λ is set to 1 (the baseline). The orange curve shows the competitive ratio across the varying instances when we use a batch prediction. Precisely, this means we use the optimal offline solution of the *first* instance and this hint is fixed for all future instances which noisily drift away from the first instance. The blue curve showcases more powerful predictions where the offline optimal of time step $t-1$ is used as the hint for time step t. We display the average values across 20 instances. We see that as the time step increases, the orange curve drifts upwards, which is intuitive as the problem instances are increasingly different. Nevertheless, the hint stays valid for many time steps as the orange curve is still below the green baseline even after many time steps. As expected, the blue curve consistently has the lowest competitive ratio as the hints are also updated. We do not shade in the standard deviation to increase the clarity of the figure but the variance of the curves similar to Figure 2a.

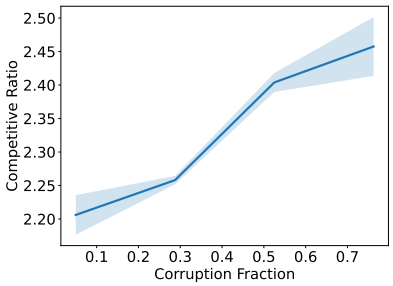
We now describe the experimental results on our graph dataset. In Figure 3a, the green curve represents not using any hints (baseline) while the orange curve shows the competitive ratio as we vary the graph instance while using the hint derived from graph #1 (λ is set to 0.1). It is shown that the hints help outperform the baseline and in addition, the hints stay accurate even if the structure of graph #9 has drifted away from that of graph #1. The online predictions, shown in blue, does marginally better than the batch version.

Figure 3b shows a similar plot as Figure 2b. We consider a replacement rate strategy where we zero out coordinates of the hint vector independently with varying probabilities. The curve shows the average of 5 trials and $\lambda = 0.1$ again. The same qualitative message as Figure 2b holds: while the corruption rate is small, we achieve a similar competitive ratio as in Figure 3a and as the corruption rate increases, there is a smooth increase in the competitive ratio.

In addition to extending and complementing the experimental results of [BMS20], we summarize our experimental results in the following points: (a) Our theory is predictive of experimental performance and



(a)



(b)

Figure 3: Figures for our time-varying graph dataset.

qualitatively validates our robustness and consistency trade offs. (b) Our algorithm framework which underlies all of our algorithm contributions is efficient to carry out and execute in practice. (c) Learning-augmented online algorithms can be applied to real world datasets varying over time such as in the analysis of graphs derived from a dynamic network.

5 Conclusion

We present the first framework for the learning-augmented online covering LP and SDP problem. As shown in table 1, for the problems without box constraints, our algorithms are $O(1/(1-\lambda)))$-consistent and $O(\log(\kappa n/\lambda))$-robust; for the problems with box constraints, our algorithms are $O(1/(1-\lambda)))$-consistent and $O(\log(s/\lambda))$-robust. Our algorithms not only support fractional advice but also update the variables efficiently. Our work raises several open questions.

First, for the problem without box constraints, is it possible to remove the dependency on the condition number κ in the robustness ratio? The competitive ratio of the optimal online algorithms for covering LPs [BN09a] and SDPs [EKN16] is $O(\log n)$. For the learning-augmented set cover problem where the entries in the constraint matrix are zeros and ones, the robustness ratio is $O(\log(d/\lambda))$, where the worst case for d is $d = n$. It seems that the condition number arises due to additive term D_j used for the growth rate. Another direction is to discover the trader-off lower bounds between the robustness and consistency ratio for the online covering and SDP covering problem as in [WZ20] for the ski rental and the non-clairvoyant scheduling problem.

Next, our technique can potentially be used on other online covering problems [ABC+16, SN20, GN14] for the learning-augmented extension. It would be exciting to see if the primal-dual continuous augmentation approach with well-selected growth rates can be employed for these problems.

As stated in [BMS20], another natural future direction is to design PDLA algorithms for packing LPs. For general online packing LPs, an $O(1/\log \kappa)$-competitive online solution can be obtained only if a condition number κ is known offline [BN09b], implying impossibility results without assumptions. The hope here is to study structured packing problems, e.g., load balancing [BN09a] and ad-auction revenue maximization [BJN07].

Acknowledgment

We thank Roie Levin for bringing to our attention the significantly simpler algorithm described in Appendix B.

References

[AAA+06] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. A general approach to online network optimization problems. *ACM Transactions on Algorithms (TALG)*, 2(4):640–660, 2006. 6, 9, 22, 23

[AAA+09] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. The online set cover problem. *SIAM Journal on Computing*, 39(2):361–370, 2009. 2, 3, 9, 21

[AB99] Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999. 41, 42

[ABC+16] Yossi Azar, Niv Buchbinder, TH Hubert Chan, Shahar Chen, Ilan Reuven Cohen, Anupam Gupta, Zhiyi Huang, Ning Kang, Viswanath Nagarajan, Joseph Naor, et al. Online algorithms for covering and packing problems with convex objectives. In *57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 148–157, 2016. 9, 36

[ABFP13] Yossi Azar, Umang Bhaskar, Lisa Fleischer, and Debmalya Panigrahi. Online mixed packing and covering. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 85–100, 2013. 9

[ACE+20] 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. PMLR, 2020. 8

[AGKK20] Antonios Antoniadis, Themis Gouleakis, Pieter Kleer, and Pavel Kolev. Secretary and online matching problems with machine learned advice. *Advances in Neural Information Processing Systems*, 33:7933–7944, 2020. 8

[AGKP22] Keerti Anand, Rong Ge, Amit Kumar, and Debmalya Panigrahi. Online algorithms with multiple predictions, 2022. 8

[APT22] Yossi Azar, Debmalya Panigrahi, and Noam Touitou. Online graph algorithms with predictions. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 35–66. SIAM, 2022. 8

[AW02] Rudolf Ahlswede and Andreas Winter. Strong converse for identification via quantum channels. *IEEE Transactions on Information Theory*, 48(3):569–579, 2002. 6

[BBN12] Nikhil Bansal, Niv Buchbinder, and Joseph Naor. A primal-dual randomized algorithm for weighted paging. *Journal of the ACM (JACM)*, 59(4):1–24, 2012. 9

[BJN07] Niv Buchbinder, Kamal Jain, and Joseph Seffi Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *European Symposium on Algorithms*, pages 253–264. Springer, 2007. 36

[BMRS20] Étienne Bamas, Andreas Maggiori, Lars Rohwedder, and Ola Svensson. Learning augmented energy minimization via speed scaling. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020. 8

[BMS20] Étienne Bamas, Andreas Maggiori, and Ola Svensson. The primal-dual method for learning augmented algorithms. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020. 2, 5, 6, 7, 8, 9, 22, 34, 35, 36

[BN09a] Niv Buchbinder and Joseph Naor. The design of competitive online algorithms via a primal-dual approach. *Found. Trends Theor. Comput. Sci.*, 3(2-3):93–263, 2009. 3, 4, 6, 7, 9, 36

[BN09b] Niv Buchbinder and Joseph Naor. Online primal-dual algorithms for covering and packing. *Mathematics of Operations Research*, 34(2):270–286, 2009. 2, 3, 5, 7, 8, 9, 36

[CEI+22] Justin Y. Chen, Talya Eden, Piotr Indyk, Honghao Lin, Shyam Narayanan, Ronitt Rubinfeld, Sandeep Silwal, Tal Wagner, David P. Woodruff, and Michael Zhang. Triangle and four cycle counting with predictions in graph streams. In *The Tenth International Conference on Learning Representations, ICLR*, 2022. 8, 34, 40, 41, 42

[CSVZ22] Justin Y. Chen, Sandeep Silwal, Ali Vakilian, and Fred Zhang. Faster fundamental graph algorithms via learned predictions. In *International Conference on Machine Learning, ICML*, pages 3583–3602, 2022. 34, 40, 41, 42

[DIL+21] Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Faster matchings via learned duals. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 10393–10406, 2021. 34

[DIRW20] Yihe Dong, Piotr Indyk, Ilya P. Razenshteyn, and Tal Wagner. Learning space partitions for nearest neighbor search. In *8th International Conference on Learning Representations, ICLR, 2020*, 2020. 8

[DKT+21] Ilias Diakonikolas, Vasilis Kontonis, Christos Tzamos, Ali Vakilian, and Nikos Zarifis. Learning online algorithms with distributional advice. In *Proceedings of the 38th International Conference on Machine Learning, ICML*, pages 2687–2696, 2021. 8

[EFS+22] Jon C. Ergun, Zhili Feng, Sandeep Silwal, David P. Woodruff, and Samson Zhou. Learning-augmented k-means clustering. In *The Tenth International Conference on Learning Representations, ICLR*, 2022. 8, 34, 40, 41, 42

[EIN+21] Talya Eden, Piotr Indyk, Shyam Narayanan, Ronitt Rubinfeld, Sandeep Silwal, and Tal Wagner. Learning-based support estimation in sublinear time. In *9th International Conference on Learning Representations, ICLR*, 2021. 8

[EKN16] Noa Elad, Satyen Kale, and Joseph (Seffi) Naor. Online semidefinite programming. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP*, pages 40:1–40:13, 2016. 2, 4, 5, 6, 7, 23, 29, 36

[FMMM09] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and Shan Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 117–126. IEEE, 2009. 8

[GKR00] Naveen Garg, Goran Konjevod, and Ramamoorthi Ravi. A polylogarithmic approximation algorithm for the group steiner tree problem. *Journal of Algorithms*, 37(1):66–84, 2000. 22

[GL20] Anupam Gupta and Roie Levin. The online submodular cover problem. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1525–1537. SIAM, 2020. 42

[GLQ21] Elena Grigorescu, Young-San Lin, and Kent Quanrud. Online directed spanners and steiner forests. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021)*, 2021. 3, 6, 7

[GN14] Anupam Gupta and Viswanath Nagarajan. Approximating sparse covering integer programs online. *Mathematics of Operations Research*, 39(4):998–1011, 2014. 9, 36, 42, 43

[GP19] Sreenivas Gollapudi and Debmalya Panigrahi. Online algorithms for rent-or-buy with expert advice. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, pages 2319–2327, 2019. 8

[GW95] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995. 9

[HIKV19] Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. Learning-based frequency estimation algorithms. In *7th International Conference on Learning Representations, ICLR*, 2019. 8

[ISZ21] Zachary Izzo, Sandeep Silwal, and Samson Zhou. Dimensionality reduction for wasserstein barycenter. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS*, pages 15582–15594, 2021. 40, 41, 42

[JLL+20] Tanqiu Jiang, Yi Li, Honghao Lin, Yisong Ruan, and David P. Woodruff. Learning-augmented data stream algorithms. In *8th International Conference on Learning Representations, ICLR*, 2020. 8

[KDZ+17] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30, 2017. 8

[KMMO94] Anna R. Karlin, Mark S. Manasse, Lyle A. McGeoch, and Susan S. Owicki. Competitive randomized algorithms for non-uniform problems. *Algorithmica*, 11(6):542–571, 1994. 9

[LFKF18] Mario Lucic, Matthew Faulkner, Andreas Krause, and Dan Feldman. Training gaussian mixture models at scale via coresets. *Journal of Machine Learning Research*, 18(160):1–25, 2018. 41, 42

[LK14] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014. 34

[LKF05] Jure Leskovec, Jon M. Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 177–187, 2005. 34

[LLMV20] Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online scheduling via learned weights. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1859–1877, 2020. 8

[LV18] Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, pages 3302–3311, 2018. 8

[Mit18] Michael Mitzenmacher. A model for learned bloom filters and optimizing by sandwiching. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 462–471, 2018. 8

[MNS12] Mohammad Mahdian, Hamid Nazerzadeh, and Amin Saberi. Online optimization with uncertain information. *ACM Transactions on Algorithms (TALG)*, 8(1):1–29, 2012. 8

[ND21] Kim Thang Nguyen and Christoph Dürr. Online primal-dual algorithms with predictions for packing problems. *CoRR*, abs/2110.00391, 2021. 8, 9

[PSK18] Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ML predictions. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 9684–9693, 2018. 8

[Roh20] Dhruv Rohatgi. Near-optimal bounds for online caching with machine learned advice. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1834–1845, 2020. 8

[SN20] Xiangkun Shen and Viswanath Nagarajan. Online covering with l_q-norm objectives and applications to network design. *Mathematical Programming*, 184, 2020. 9, 36, 42

[SZS+14] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR, Conference Track Proceedings*, 2014. 2

[VB96] Lieven Vandenberghe and Stephen P. Boyd. Semidefinite programming. *SIAM Rev.*, 38(1):49–95, 1996. 2

[WX06] Avi Wigderson and David Xiao. Derandomizing the aw matrix-valued chernoff bound using pessimistic estimators and applications. *ECCC TR06-105*, 2006. 6

[WZ20] Alexander Wei and Fred Zhang. Optimal robustness-consistency trade-offs for learning-augmented online algorithms. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS*, 2020. 8, 36

A Learnability of Predictor

In this section, we present formal learning bounds for learning a good predictor for the covering formulation of online semidefinite programming, which includes learning a good predictor for online fractional linear programming. Specifically, we prove that a good predictor can be efficiently learned if the problem instances are drawn from a particular distribution, in the style of the probably approximately correct (PAC) learning framework for data-driven algorithm design. Similar ideas were used by [EFS+22, ISZ21, CEI+22, CSVZ22] to show efficient learners for predictors for k-means clustering and triangle counting. Thus while the results of this section use ideas of prior papers, they are still meaningful as it provides an "end-to-end" recipe for designing learning-augmented online algorithms. The prior part of this paper designs online algorithms which leverage predictions and thus, the natural follow up questions are how efficiently one can learn these hint from data. Therefore this learnability section precisely addresses these questions and makes our results complete.

Suppose there exists an underlying distribution $\mathcal{D}$ that generates independent covering semidefinite programming or covering linear program instances, capturing the case where similar problem instances are being solved. Note that this setting also mirrors some of our experiments, specifically in the case of our online covering datasets which are derived from an underlying dataset (an internet router graph), sampled across time. We would like to efficiently learn a good predictor f from a family of functions $\mathcal{F}$, where the input to f is an covering SDP $\mathcal{S}$ and the output is a weight vector. Each covering SDP instance can be written in terms of $d := O(nm^2)$ variables, so we assume that each input instance $\mathcal{S}$ is encoded as a vector in $\mathbb{R}^d$. The output of f is then an n-dimensional vector, so that each coordinate $i \in [n]$ represents the associated weight of the matrix A_i selected in the covering SDP.

To quantify the quality of each function f, we consider a loss function $L : f \times \mathcal{S} \to \mathbb{R}$ that outputs the cost of corresponding SDP cover. For example, f can output a set of matrix weights given an instance $\mathcal{S}$. If the matrix weights are not feasible, then the loss function may output ∞. Otherwise if the matrix weights are feasible, then the loss function may output the corresponding cost, defined by the cost vector that is also given in $\mathcal{S}$.

We would like to learn the best function $f \in \mathcal{F}$ that minimizes the objective:

$$\mathbb{E}_{\mathcal{S}\sim\mathcal{D}}[L(f, \mathcal{S})]. \tag{21}$$

Let f^* denote the minimizer of the above objective, i.e., the optimal function in $\mathcal{F}$, so that

$$f^* = \operatorname{argmin} \mathbb{E}_{\mathcal{S}\sim\mathcal{D}}[L(f, \mathcal{S})].$$

We assume that for each SDP instance $\mathcal{S}$ and each $f \in \mathcal{F}$ that $f(\mathcal{S})$ and $L(f, \mathcal{S})$ can be computed in time that is a (small) polynomial in n and d, which we denote by $T(n, d)$.

We show that there exists an efficient algorithm that outputs a function $\hat{f} \in \mathcal{F}$ whose loss is within an additive ε of the loss of the optimal function f^*.

Theorem A.1. *There exists an algorithm that uses* poly $\left(T(n,d), \frac{1}{\varepsilon}\right)$ *samples and outputs a function* $\hat{f}$ *that satisfies with probability at least* $\frac{9}{10}$,

$$\mathbb{E}_{\mathcal{S}\sim\mathcal{D}}[L(\hat{f}, \mathcal{S})] \leq \mathbb{E}_{\mathcal{S}\sim\mathcal{D}}[L(f^*, \mathcal{S})] + \varepsilon,$$

where $f^* = \operatorname{argmin} \mathbb{E}_{\mathcal{S}\sim\mathcal{D}}[L(f, \mathcal{S})]$.

Theorem A.1 shows that only a small number of samples are needed to have a good probability of learning an approximately-optimal function $\hat{f}$, in a typical PAC-style bound. Specifically, the algorithm to compute $\hat{f}$ is simply the empirical risk minimizer, i.e., the algorithm minimizes the empirical loss after an appropriate number of samples are drawn. To prove correctness of Theorem A.1, we first define the notion of pseudo-dimension for a function class, which generalizes the more familiar VC dimension to real functions, e.g., Definition 9 in [LFKF18].

Definition A.2 (Pseudo-Dimension)**.** Let $\mathcal{X}$ be a ground set and $\mathcal{F}$ be a set of functions from $\mathcal{X}$ to the interval $[0, 1]$. Let $U = \{x_1, \cdots, x_n\} \subset \mathcal{X}$ be a fixed set, $R = \{r_1, \cdots, r_n\}$ be a fixed set of real numbers with $r_i \in [0, 1]$ for all $i \in [n]$, and $f \in \mathcal{F}$ be a fixed function. The set $U_f = \{x_i \in U \mid f(x_i) \geq r_i\}$ is called the induced subset of U formed by f and R. The set U with associated values R is shattered by $\mathcal{F}$ if $|\{U_f \mid f \in \mathcal{F}\}| = 2^n$. The *pseudo-dimension* of $\mathcal{F}$ is the cardinality of the largest shattered subset of $\mathcal{X}$ (or ∞).

Let $\mathcal{G}$ be the class of functions in $\mathcal{F}$ composed with L, so that

$$\mathcal{G} := \{L \circ f : f \in \mathcal{F}\}.$$

Without loss of generality, we normalize the range of the loss function L to $[0, 1]$. Then the performance of the empirical risk minimization and the number of necessary samples can be related to the pseudo-dimension using standard bounds as follows:

Theorem A.3. *[AB99] Let $\mathcal{D}$ be a distribution over problem instances $\mathcal{S}$ and $\mathcal{G}$ be a class of functions $g : \mathcal{S} \to [0, 1]$ with pseudo-dimension $d_{\mathcal{G}}$. Consider t i.i.d. samples $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_t$ from $\mathcal{D}$. There exists a universal constant C, such that for any $\varepsilon > 0$, if $t \geq \frac{C \cdot d_{\mathcal{G}}}{\varepsilon^2}$, then with probability at least $\frac{9}{10}$,*

$$\left| \frac{1}{t} \sum_{i=1}^{t} g(\mathcal{S}_i) - \mathbb{E}_{\mathcal{S}\sim\mathcal{D}}\, g(\mathcal{S}) \right| \leq \varepsilon$$

for all $g \in \mathcal{G}$.

From the triangle inequality, we then have the following corollary.

Corollary A.4. *Let $t \geq \frac{C \cdot d_{\mathcal{G}}}{\varepsilon^2}$, where C is the universal constant from Theorem A.3. Consider a set of t independent samples $\mathcal{S}_1, \ldots, \mathcal{S}_t$ from $\mathcal{D}$ and let $\hat{g}$ be a function in $\mathcal{G}$ that minimizes $\frac{1}{t}\sum_{i=1}^{t} g(\mathcal{S}_i)$. Then with probability at least $\frac{9}{10}$,*

$$\mathbb{E}_{\mathcal{S}\sim\mathcal{D}}[\hat{g}(\mathcal{S})] \leq \mathbb{E}_{\mathcal{S}\sim\mathcal{D}}[g^*(\mathcal{S})] + 2\varepsilon.$$

Hence, the main question is whether we can bound the pseudo-dimension of the given function class $\mathcal{G}$. To that end, we first observe that the pseudo-dimension can be related to the VC dimension of a related class of threshold functions. In particular, this relationship has been instrumental in showing a number of existing learning bounds, e.g., [LFKF18, ISZ21, EFS+22, CEI+22, CSVZ22].

Lemma A.5 (Pseudo-dimension to VC dimension, Lemma 10 in [LFKF18]). *For any $g \in \mathcal{G}$, let $B_g(x, y) = sgn(g(x) - y)$, i.e., the indicator function of the region on or below the graph of g. Then the pseudo-dimension of $\mathcal{G}$ is equivalent to the VC dimension of the subgraph class $B_{\mathcal{G}} = \{B_g \mid g \in \mathcal{G}\}$.*

We can also relate the VC dimension of a given function class to the computational complexity of the function class, i.e., the time complexity of computing a function in the class.

Lemma A.6 (Theorem 8.14 in [AB99]). *Let $h : \mathbb{R}^a \times \mathbb{R}^b \to \{0, 1\}$ define the class*

$$\mathcal{H} = \{x \to h(\theta, x) : \theta \in \mathbb{R}^a\}.$$

Suppose that h can be computed by an algorithm that takes as input the pair $(\theta, x) \in \mathbb{R}^a \times \mathbb{R}^b$ and returns $h(\theta, x)$ after no more than t of the following operations:

- *arithmetic operations $+, -, \times$, and $/$ on real numbers,*
- *jumps conditioned on $>, \geq, <, \leq, =$, and $=$ comparisons of real numbers, and*
- *output $0, 1$.*

Then the VC dimension of $\mathcal{H}$ is at most $O(a^2t^2 + t^2a \log a)$.

We can now prove Theorem A.1 by roughly instantiating Lemma A.6 using the computational complexity of any function in the function class $\mathcal{G}$.

Proof of Theorem A.1. The proof essentially follows from the proofs of similar theorems in the sample complexity bound sections of the papers [LFKF18, ISZ21, EFS+22, CEI+22, CSVZ22] but we nonetheless repeat it here for completeness. By Theorem A.3 and Corollary A.4, we observe that it suffices to bound the pseudo-dimension of the class $\mathcal{G} = L \circ \mathcal{F}$. By Lemma A.5, the pseudo-dimension of $\mathcal{G}$ is equivalent to the VC dimension of threshold functions defined by $\mathcal{G}$. Moreover by Lemma A.6, the VC dimension of the threshold functions defined by $\mathcal{G}$ is polynomial in the complexity of computing a member of the function class.

Hence, Lemma A.6 implies that the VC dimension of $B_{\mathcal{G}}$ defined in Lemma A.5 is polynomial in the number of arithmetic operations needed to compute the threshold function associated to some $g \in \mathcal{G}$, which is polynomial in $T(n, d)$ by our definition. Therefore, the pseudo-dimension of $\mathcal{G}$ is also polynomial in $T(n, d)$ and the desired claim follows. □

B A Simple Learning-Augmented Online Algorithm

In this section, we revisit a simple and possibly folklore learning-augmented online algorithm for online covering LPs. Its robustness ratio is asymptotically as good as the state-of-the-art online algorithm and its consistency ratio is also asymptotically as good as the advice if the advice is feasible. We recall that a similar approach also applies to a wide range of learning-augmented online minimization problems, including online SDPs, online LPs and SDPs with box constraints, online submodular cover [GL20], and online covering with ℓ_q-norm objectives [SN20].

The idea of the algorithm is to maintain two candidate solutions, the advice x' and the solution obtained by a state-of-the-art online algorithm [GN14]. Let $\mathcal{O}$ be a state-of-the-art online algorithm and let $x^{(i)}$ be its solution obtained at round i. In the beginning, $x^{(0)}$ is initialized according to the online algorithm $\mathcal{O}$. We follow the online algorithm $\mathcal{O}$ until its objective $c^T x^{(i)}$ is larger than the objective of the advice $c^T x'$. Once the advice is infeasible, we ignore the advice and use the online algorithm. Transitioning either from the online algorithm to the advice or from the advice to the online algorithm pays only a constant factor. Once we transition, to maintain monotonicity of the online solution, we pick the coordinate-wise maximum between the advice and the solution by $\mathcal{O}$. The algorithm works as follows.

Algorithm 9 A Simple Algorithm for Learning-Augmented Online Covering LPs

1: $i_a \leftarrow \infty$. ▷ i_a is the round that we transition to the advice
2: **for** $i = 1, 2, \ldots$ **do** ▷ each arriving row or constraint
3: Update A by adding a new row i.
4: Run $\mathcal{O}$ for round i and obtain $x^{(i)}$.
5: **if** $Ax' \geq \mathbf{1}$ and $c^T x' \geq c^T x^{(i)}$ **then** ▷ use the online solution
6: $x \leftarrow x^{(i)}$.
7: **if** $Ax' \geq \mathbf{1}$ and $c^T x' < c^T x^{(i)}$ and $i < i_a$ **then** ▷ transition to the advice
8: $i_a \leftarrow i$. ▷ (only once at round i_a)
9: **for** $j \in [n]$ **do**
10: $x_j \leftarrow \max\{x'_j, x_j^{(i-1)}\}$.
11: **if** $Ax' \geq \mathbf{1}$ does not hold **then** ▷ use only the online solution
12: **if** $i < i_a$ **then** ▷ no transition to advice
13: $x \leftarrow x^{(i)}$.
14: **if** $i \geq i_a$ **then** ▷ has transitioned to advice
15: **for** $j \in [n]$ **do** ▷ transition back to the online solution
16: $x_j \leftarrow \max\{x'_j, x_j^{(i)}\}$.

Theorem B.1. *For the learning-augmented online covering LP problem, there exists an online algorithm that generates x such that*

$$c^T x \leq \min\{O(c^T x'), O(\log k)\mathit{OPT}\}$$

when x' is feasible, and $c^T x \leq O(\log k)\mathit{OPT}$ when x' is infeasible. Here, $k \leq n$ is the row sparsity of A, i.e., the maximum number of non-zero entries of each row.

Proof. We use Algorithm 9 with the $O(\log k)$-competitive online algorithm for $\mathcal{O}$.

Theorem B.2 ([GN14]). *There exists an $O(\log k)$-competitive algorithm for the online covering LP problem.*

Algorithm 9 considers four possible cases.

1. The advice is feasible but $\mathcal{O}$ is better (line 5).
2. The advice is feasible and it is better than $\mathcal{O}$ (line 7 or x is not updated).
3. The advice is infeasible and never used (line 12).
4. The advice is infeasible and was used when it was feasible (line 14).

Clearly, Algorithm 9 always returns a feasible solution x and x is updated in a non-decreasing manner. We first consider the case when x' is feasible. For the first case, we never use the advice, so

$$c^T x \leq O(\log k)\mathsf{OPT} \leq \min\{O(c^T x'), O(\log k)\mathsf{OPT}\}.$$

For the second case, either the advice becomes better than the online solution at round i or x is not updated because the advice becomes better than the online solution at a previous round $i_a < i$. For the previous case, by the design of Algorithm 9, we have that

$$c^T x \leq c^T(x' + x^{(i-1)}) \leq 2c^T x' \leq \min\{O(c^T x'), O(\log k)\mathsf{OPT}\}.$$

For the later case, x is not updated after round i_a so the claim still holds.

The remaining cases are when x' is infeasible. For the third case, we never use the advice, so $c^T x \leq O(\log k)\mathsf{OPT}$. For the last case, let OPT_i denote the value of OPT at round i. We know that the advice was

feasible at some previous round $i' \geq i_a$ but became infeasible at round $i' + 1 \leq i$. Round i' belongs to the second case, so at that round

$$c^T x \leq 2c^T x'.$$

In round $i' + 1$, we transition back from the advice to the online algorithm, so

$$c^T x \leq 2c^T x' + O(\log k)\mathsf{OPT}_{i'+1} \leq \min\{O(c^T x'), O(\log k)\mathsf{OPT}_{i'+1}\} \leq O(\log k)\mathsf{OPT}_{i'+1}.$$

In round i, $c^T x'$ does not change but OPT changes. By the property of the online algorithm $\mathcal{O}$, we always have

$$c^T x \leq O(\log k)\mathsf{OPT}_i.$$

□