LACS: Learning-Augmented Algorithms for Carbon-Aware Resource Scaling with Uncertain Demand

Roozbeh Bostandoost
University of Massachusetts Amherst
USA

Adam Lechowicz
University of Massachusetts Amherst
USA

Walid A. Hanafy
University of Massachusetts Amherst
USA

Noman Bashir
Massachusetts Institute of Technology
USA

Prashant Shenoy
University of Massachusetts Amherst
USA

Mohammad Hajiesmaili
University of Massachusetts Amherst
USA

ABSTRACT

Motivated by an imperative to reduce the carbon emissions of cloud data centers, this paper studies the online carbon-aware resource scaling problem with unknown job lengths (OCSU) and applies it to carbon-aware resource scaling for executing computing workloads. The task is to dynamically scale resources (e.g., the number of servers) assigned to a job of unknown length such that it is completed before a deadline, with the objective of reducing the carbon emissions of executing the workload. The total carbon emissions of executing a job originate from the emissions of running the job and excess carbon emitted while switching between different scales (e.g., due to checkpoint and resume). Prior work on carbon-aware resource scaling has assumed accurate job length information, while other approaches have ignored switching losses and require carbon intensity forecasts. These assumptions prohibit the practical deployment of prior work for online carbon-aware execution of scalable computing workload.

We propose LACS, a theoretically robust, learning-augmented algorithm that solves OCSU. To achieve improved practical average-case performance, LACS integrates machine-learned predictions of job length. To achieve solid theoretical performance, LACS extends the recent theoretical advances on online conversion with switching costs to handle a scenario where the job length is unknown. Our experimental evaluations demonstrate that, on average, the carbon footprint of LACS lies within 1.2% of the online baseline that assumes perfect job length information and within 16% of the offline baseline that, in addition to the job length, also requires accurate carbon intensity forecasts. Furthermore, LACS achieves a 32% reduction in carbon footprint compared to the deadline-aware carbon-agnostic execution of the job.

CCS CONCEPTS

• **Theory of computation** → **Online algorithms**; • **Social and professional topics** → **Sustainability**.

E-Energy '24, June 04–07, 2024, Singapore, Singapore

ACM ISBN 979-8-4007-0480-2/24/06
https://doi.org/10.1145/3632775.3661942

KEYWORDS

Sustainable computing, online algorithms, resource scaling

ACM Reference Format:
Roozbeh Bostandoost, Adam Lechowicz, Walid A. Hanafy, Noman Bashir, Prashant Shenoy, and Mohammad Hajiesmaili. 2024. LACS: Learning-Augmented Algorithms for Carbon-Aware Resource Scaling with Uncertain Demand. In *The 15th ACM International Conference on Future and Sustainable Energy Systems (E-Energy '24), June 04–07, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3632775.3661942

1 INTRODUCTION

The exponential growth in computing demand and the resulting energy consumption has enhanced focus on its climate and sustainability implications [4, 36, 37, 43]. The focus has been magnified since the widespread adoption of generative artificial intelligence tools, e.g., ChatGPT [29]. Key stakeholders, including policymakers and end users, are trying to create direct incentives, through caps or taxes [10] on carbon emissions, and indirect incentives, through social pressure, to curb the climate impact of this unprecedented demand. In response, researchers and other stakeholders in computing are trying to reduce the carbon footprint during its various lifecycle stages, including manufacturing [18, 49], operations [3, 20, 41, 53], and end-of-life [17, 48]. While computing's carbon footprint at all stages are important to address, this paper focuses on the operational carbon footprint arising from using electricity to run computing workloads, as it contributes significantly to computing's total carbon footprint [18].

Beyond improving the algorithmic efficiency of computing workloads and the energy efficiency of its hardware, computing's operational footprint can be reduced by enhancing the carbon efficiency of grid-supplied electricity (kilowatt-hours of energy produced per unit of carbon emissions) [4]. One approach is to use low-carbon energy sources, such as solar, wind, and nuclear, for electricity generation. However, as pathways to 100% renewable energy adoption remain challenging and costly [2, 9], this approach may not entirely eliminate electricity's carbon emissions for all the locations in the near future [21]. A complementary approach is to improve the *effective* carbon efficiency of the energy *used* for computing by running flexible computing workloads when and where low-carbon energy is available. Prior work has proposed leveraging computing workloads' spatiotemporal flexibility [15, 23, 26, 44, 53] and resource elasticity [19, 20, 25, 41] to reduce their carbon footprint.

In this work, we focus on exploiting resource elasticity for carbon-aware execution of scalable computing workloads with unknown job lengths and future carbon intensity. The carbon-aware resource scaling work requires determining the scale factor, i.e., the number of cores or servers, at each time step before the job deadline while considering the scalability properties of the job. Initial work on carbon-aware resource scaling by Hanafy et al. [20] leverages carbon intensity forecasts and develops an offline optimal approach to determine the best scale factor for a job at each time step before the deadline. The authors ignore switching overhead and assume the deadline is provided at job submission time. Lechowicz et al. [25] introduce and study an online class of problems motivated by carbon-aware resource scaling and electric vehicle (EV) charging applications. The proposed algorithms can be used to determine the optimal scale factor without requiring carbon intensity forecasts while considering the energy inefficiencies in resource scaling through a convex cost function, which is revealed online. However, a key drawback for both studies is that they assume each job's length (i.e., the amount of work to be done) is known.

Estimating the duration of a job remains a challenging problem in cluster and cloud computing. The unavailability of data on job attributes, lack of diversity in the available data, variations in the characteristics of the jobs submitted to the cluster over time, and skewed distribution of users submitting the jobs means that job length predictions, even when using machine learning techniques, remain highly inaccurate and cannot be used for scheduling purposes [24]. Ambati et al. [1] showed that job length estimates at job submission time can have more than 140% mean absolute percentage error. The inaccuracy can be further amplified as the properties of the job or the hardware it runs on change across different runs. As a result, practical algorithms need to work without assuming that accurate job length is available. In this work, we assume that the job lengths are unknown and only the lower- and upper-bounds on job length are available. This is a reasonable assumption as classifying a job as short or long tends to be highly accurate [1, 59].

The existing theoretical literature that studies similar problems crucially does not consider uncertainty in the job length [16, 25, 26, 28, 47] (i.e., the job length is precisely known to the algorithm). In perhaps the closest setting to carbon-aware resource scaling, Lechowicz et al. [25] presents the online conversion with switching costs (OCS) problem. Uncertainty about the job length breaks many of the assumptions in OCS; under a deadline and without precise information on job length, the algorithm will either run too little of the job at a low carbon intensity or run too much of the job at a high carbon intensity. Without even approximate knowledge of job length (i.e., lower- and upper-bounds on job lengths) and under a deadline, the existing algorithms, e.g., for OCS, may not complete the job by the deadline and thus fail to provide worst-case guarantees.

It is worth noting that providing worst-case guarantees is an important consideration for algorithms solving this problem. In-production resource managers such as Borg [51] and Resource Central [12], prefer deploying techniques with safety guarantees [39]. Techniques which fully rely on machine learning (ML) perform well in the average-case, but can result in extremely poor outcomes in the worst-case (e.g., when presented with out of distribution data), making them undesirable for production deployment [24].

Contributions. This paper proposes LACS, a learning-augmented algorithm for online carbon-aware resource scaling with unknown job lengths (OCSU) that uses ML predictions of job lengths, which are potentially inaccurate, for resource scaling. We then analyze the theoretical performance of LACS using the framework of competitive analysis [6], and its learning-augmented variants [30, 38]. We also evaluate the practical performance of LACS using real-world data traces on an extensive set of experimental scenarios.

(1) *Theoretical analysis of* LACS*: Bounded robustness and consistency.* The theoretical analysis of LACS leverages and advances the emerging framework of robustness-consistency for learning-augmented online algorithms [30, 38] and the recent competitive results for OCS [25], which tackle a simplified OCSU with known job lengths. We take multiple algorithmic steps to achieve bounded competitive guarantees for LACS. First, we consider two extreme scenarios of the "Ramp-On Ramp-Off" (RORO) framework proposed by Lechowicz et al. [25] to design *robust* baseline algorithms of assuming actual job lengths equal to the given lower and upper bound values of job lengths. Then, using these two extremes as the baseline, we introduce an additional layer by integrating a job length predictor, and finally, LACS leverages these robust baseline algorithms, combined with potentially inaccurate predictions of the job length, to achieve improved consistency in the average case (i.e., when predictions are of high quality), while retaining worst-case guarantees (i.e., robustness given by the baselines).
(2) *Extensive trace-driven experiments.* We then evaluate the performance of LACS against state-of-the-art methods in carbon-aware scheduling and online scheduling literature using three years of real carbon intensity traces from Electricity Maps [32], using an extensive range of experimental scenarios. In a set of representative experiments, we demonstrate that, on average, the carbon footprint of LACS lies within 1.2% of the online baseline that assumes perfect job length information and within 16% of the offline baseline that also requires accurate carbon intensity forecasts. LACS achieves a 32% reduction in carbon footprint compared to the deadline-aware carbon-agnostic job execution.

2 PROBLEM STATEMENT

In the following, we will consider a server cluster that is used to run batch jobs. We introduce the online carbon-aware resource scaling problem with unknown job lengths (OCSU), where the goal is to complete a job with total length c (where c is unknown) while minimizing its overall carbon emissions to complete the job. At each time step $t \in [T]$, a convex cost function $g_t(\cdot)$ arrives, which is a combination of both the time-varying carbon intensity (e.g., of the electricity grid) and the job's time-varying scaling profile (e.g., how parallelizable the job is). In response, the algorithm must choose the amount of server resources x_t that will be given to the job in time step t, where $x_t \in [0, d_t]$, which produces carbon emissions given by $g_t(x_t)$. Here, d_t is the maximum amount of job that can be scheduled for the job at time t (rate constraint). Intuitively, we assume that $g_t(0) = 0$ for any cost function (i.e., completing none of the job does not emit any carbon), and $g_t(x) \geq 0$ for any valid $x > 0$.

Table 1: Summary of Notations

Notation	Definition
c, $c_{\min}$, $c_{\max}$	Actual job length, minimum job length, maximum job length
$\hat{c}$	Job length prediction
β	Switching emissions coefficient
CI_t	Carbon intensity (e.g., in $gCO_2eq./kWh$) at time t
E	Energy used (e.g. in kWh) by one unit of threads/cores/servers
T	Deadline: Maximum time (e.g., 24 hours) the job is allowed to run after being submitted
$g_t(\cdot)$	Convex cost function that arrives at time t
x_t	Amount of the job that is scheduled to be done at time t
$w^{(t)}$	Total amount of the job that has been completed up to time t
r_t	Maximum resource available at time t
d_t	Maximum amount of the job that can be scheduled at time t

Whenever the allocation decision changes in adjacent time steps, it incurs extra carbon emissions caused by switching denoted by $\beta|x_t - x_{t-1}|$. For the model, we let $x_0 = 0$ and $x^{T+1} = 0$, which require the algorithm to incur *some* switching carbon emissions to "turn on" and "turn off", respectively. The parameter β can be interpreted as a linear coefficient that charges the algorithm proportionally to the amount of scaling between consecutive time steps, based on, e.g., the carbon emitted due to overhead of changing the resource allocation or checkpointing/resuming the job.[1]. In summary, the offline version of OCSU is formalized as:

$$\text{OCSU}: \min_{\{x_t\}_{t\in[T]}} \underbrace{\sum_{t=1}^{T} g_t(x_t)}_{\text{Execution carbon emissions}} + \underbrace{\sum_{t=1}^{T+1} \beta|x_t - x_{t-1}|}_{\text{Switching carbon emissions}}, \quad (1)$$

$$\text{s.t.}, \quad \underbrace{\sum_{t=1}^{T} x_t \geq c,}_{\text{Job completion constraint}} \quad x_t \in [0, d_t], \ \forall t \in [T]. \quad (2)$$

In this paper, we focus on designing algorithms for the online version of this problem, where the algorithm must choose an irrevocable x_t at each time step without knowledge of future cost functions or the total job length c. Each cost function $g_t(\cdot)$ is revealed online at the start of time step t, and the actual job length c is revealed when the constraint in Equation 2 is satisfied (i.e., the job has been finished). We note that OCSU builds on the existing formulation of *online conversion with switching costs* (OCS), introduced by Lechowicz et al. [25]. The minimization variant of OCS is a special case of OCSU where the online algorithm has perfect knowledge of the actual job length c. The core notations are summarized in Table 1.

Details of the cost function. To formalize the definition of the cost functions $g_t(\cdot)$, let $h_t(\cdot)$ denote the scaling profile of the job at time t. For a given s, as the number of allocated resources (e.g., cores or servers), $h_t(s)$ provides the throughput (e.g., amount of work done) x at time t. Naturally, $h_t(\cdot)$ is a concave function since adding resources has diminishing returns even in highly parallel computing workloads. Note that $h_t^{-1}(x)$ (i.e., the inverse of the scaling function) maps a throughput x to the necessary amount of allocated resources s at time t. There is a *carbon emissions* associated with the resource allocation amount of s. Let E denote the energy used (e.g., in kWh) by one unit of resource, and let CI_t denote the carbon intensity (e.g., in $gCO_2eq./kWh$) at time t. Then, the cost function $g_t(\cdot)$ of OCSU is defined as:

$$g_t(x) = \mathrm{CI}_t \times E \times h_t^{-1}(x), \quad t \in [T].$$

Since the scaling profile $h_t(\cdot)$ is known, the primary *unknown* quantity which makes $g_t(\cdot)$ an online input is the unknown carbon intensity CI_t. Furthermore, considering that the available resources constrain the quantity of job that can be scheduled in each time slot, we introduce r_t to represent the maximum resources available at a given time. Consequently, the maximum amount of the job that can be scheduled at time t, denoted as d_t (maximum rate), equals $h_t(r_t)$.

Assumptions. We make the following assumptions in the paper.

- *Assumption 1.* Although the job length c is unknown, we assume that the value of c is bounded between a minimum and a maximum length $c_{\min}$ and $c_{\max}$, i.e., $c \in [c_{\min}, c_{\max}]$. Without loss of generality, we assume $c_{\min} = 1$, which further gives that $c_{\max} > c_{\min} = 1$.
- *Assumption 2.* We assume that the derivatives of the cost function are bounded, i.e., $L \leq dg_t/dx_t \leq U \ \forall t \in [T]$ on the interval $x_t \in [0, d_t]$, where L and U are known positive constants. This is a necessary assumption for any competitive algorithm, as shown in [16, 47, 58]; otherwise, no online algorithm can achieve a bounded competitive ratio.
- *Assumption 3.* The switching emissions coefficient β is known to the algorithm, and is bounded within an interval ($\beta \in [0, (U-L)/2)$), as in [25]. If β exceeds $(U-L)/2$, any competitive algorithm should only consider the excess emissions due to switching because the overhead is very large; hence the decision-making becomes trivial.
- *Assumption 4.* OCSU requires the algorithm to complete the entire job before the sequence ends at "deadline" T. If the scheduler has completed $w^{(j)}$ amount of the job at time j, a *compulsory execution* begins whenever $(T - j - 1) < (c - w^{(j)})$ (i.e., when the remaining time steps are barely enough to complete the job). During this compulsory execution, a carbon-agnostic algorithm takes over and runs the job with the maximum available resources in the remaining time steps. Although c is unknown to the algorithm, for modeling purposes, we assume that the algorithm will begin this compulsory execution when the remaining steps are sufficient to fulfill the worst-case job length, which is given by $(c_{\max} - w^{(j)})$.
- *Assumption 5.* In an application such as carbon-aware resource scaling, the deadline T is typically known in advance. Our algorithms, however, do not require this assumption to be true. If T is unknown, we assume that the algorithm is given a signal to indicate that the deadline is coming up and that compulsory execution should begin to finish a job with worst-case size $c_{\max}$.
- *Assumption 6.* We assume that the job execution time horizon has sufficient *slackness*, i.e., the compulsory execution does not make up a large fraction of the sequence – otherwise, the problem is trivial. Formally, we have that the earliest time step j' at which the compulsory execution begins (i.e., the first time step such that $(T - j' - 1) < c_{\max}$) is $j' \gg 1$, which implies that T is sized appropriately for the job. This assumption is reasonable in practice, since if T is small or c is large, the job's *temporal flexibility* will be low, so even a solution with perfect knowledge of future carbon

[1]We note that while OCSU assumes the emissions to "turn on" is equivalent to the cost to "turn off" (i.e., the switching emissions are symmetric), it can be extended to cases where the switching emissions are time-varying or asymmetric by letting $\beta|x_t - x_{t-1}|$ be an upper bound on the actual switching emissions, as discussed in [25]

Algorithm 1 Online ramp-on, ramp-off (RORO) algorithm [25]

1: **input:** pseudo-cost threshold $\phi(w)$
2: **initialization:** initial decision $x_0 = 0$, initial progress $w^{(0)} = 0$;
3: **while** cost function $g_t(\cdot)$ is revealed and $w^{(t-1)} < c$ **do**
4: solve **pseudo-cost minimization problem** to obtain decision x_t,

$$x_t = \underset{x \in [0, \min(1-w^{(t-1)}, d_t)]}{\arg\min} \; g_t(x) + \beta|x - x_{t-1}| - \int_{w^{(t-1)}}^{w^{(t-1)}+x} \phi(u)du. \quad (3)$$

5: update the progress $w^{(t)} = w^{(t-1)} + x_t$;

intensity values will be unable to take advantage of time-varying carbon intensity to reduce emissions.

Competitive analysis. We tackle OCSU from the perspective of competitive analysis, where the objective is to design an online algorithm that maintains a small *competitive ratio* [7], defined as:

Definition 2.1 (Competitive Ratio). *We denote* $\mathtt{OPT}(\mathcal{I})$ *as the offline optimum on the input* $\mathcal{I}$*, and* $\mathtt{ALG}(\mathcal{I})$ *represents the profit obtained by an online algorithm (*ALG*) on that input. Formally, letting* Ω *denote the set of all possible inputs, we say that* ALG *is* η*-competitive if the following holds:* $\mathsf{CR} = \max_{\mathcal{I} \in \Omega} \mathtt{ALG}(\mathcal{I})/\mathtt{OPT}(\mathcal{I}) = \eta$*. Observe that* CR *is greater than or equal to one. The smaller it is, the closer the algorithm is to the optimal solution.*

Learning-augmented competitive algorithms. In the nascent literature on learning-augmented algorithms [30, 38], algorithms are evaluated through the metrics of *consistency* and *robustness*. Intuitively, these quantities measure how close a learning-augmented algorithm's solution is to that of the offline optimal solution when the prediction is accurate (consistency) and how far an algorithm's solution can be from the optimal solution in the worst case when the prediction is erroneous (robustness).

Definition 2.2 (Consistency and Robustness). *Formally, an algorithm is* b*-consistent if it is* b*-competitive with respect to an accurate prediction and* r*-robust if it is* r*-competitive regardless of the quality of the prediction.*

3 ALGORITHM DESCRIPTIONS

In this section, we introduce LACS, a **L**earning-**A**ugmented **C**arbon-aware Resource **S**caling algorithm that solves OCSU. To achieve the best of both worlds on satisfactory practical performance and theoretical worst-case guarantees, LACS integrates *predictions of the job length* into its operation by combining the decisions of an algorithm that assumes the prediction is correct with the decisions of two competitive baselines. By combining these strategies, LACS can improve its performance significantly when the predictions are accurate while maintaining worst-case competitive guarantees. Below, we start by reviewing approaches from prior work that inform our design of the competitive baselines.

3.1 Algorithmic Background

The competitive baselines we consider in the next section build on prior work, specifically the "ramp-on, ramp-off" (RORO) framework proposed by [25] that achieves the optimal competitive ratio for OCS, as a simplified version of OCSU that assumes job length is known a priori to the online algorithm. In the RORO framework [25], whenever an input arrives online, the algorithm solves a *pseudo-cost minimization* problem to determine the amount of job to run at time t (denoted by $x_t \in [0, d_t]$). The progress $w^{(t)}$ denotes the fraction of the total job that has been completed up to time t. This pseudo-cost minimization design generalizes the concept of threshold-based algorithm design – at each time step, when a cost function arrives, the pseudo-cost of a particular decision x is defined as the actual carbon emissions of running x amount of the job (including both the execution and the switching emissions), minus a threshold value which describes the exact amount which should be allocated to maintain a certain competitive ratio.

This pseudo-cost acts as an incentive to prevent the algorithm from "waiting too long" to run the job. Intuitively, if an algorithm naively minimizes the cost function g_t at each time step (resulting in decisions $x_t = 0$ for all $t \in [T]$), it will be required to complete the entire job during compulsory execution during a potentially bad period for carbon intensity. The pseudo-cost minimization provides a framework that balances the extreme options of allocating "too much" early on and waiting indefinitely. Whenever the carbon intensity is "attractive enough," the pseudo-cost minimization finds the best decision that allocates just enough resources given the current carbon intensity to maintain competitiveness. In the setting where the job length is known, we summarize the RORO algorithm in Algorithm 1.

To define this pseudo-cost minimization problem, the authors in [25] define a *dynamic threshold function*, which essentially defines the highest carbon intensity deemed acceptable by RORO at time t. We note that in OCS with known job lengths, c is defined to be 1 (without loss of generality). According to [25, Definition 3.1], the dynamic threshold for OCS, for a job with length c, and for any progress $w \in [0, c]$ is defined as:

$$\phi_{\mathtt{OCS}}(w) = U - \beta + \left(\frac{U}{\alpha} - U + 2\beta\right) \exp\left(\frac{w}{c\alpha}\right), \quad (4)$$

where α is the competitive ratio defined as:

$$\alpha = \left[W\left[\left(\frac{2\beta}{U} + \frac{L}{U} - 1\right)\exp\left(\frac{2\beta}{U} - 1\right)\right] - \frac{2\beta}{U} + 1\right]^{-1}. \quad (5)$$

In the above equation, $W(\cdot)$ is the Lambert W function, defined as the inverse of $f(y) = ye^y$ [11]. Note that it is well-known that $W(x) \sim \ln(x)$ [22]. Given this definition of α, note that $\phi_{\mathtt{OCS}}(\cdot)$ is monotonically decreasing on the interval $w \in [0, c]$.

3.2 LACS: A Learning-augmented Algorithm for Carbon-aware Resource Scaling

In this section, we describe the design of LACS, which uses predictions of the actual job length to significantly improve average-case performance (consistency) without losing worst-case guarantees (robustness). We first introduce two baseline competitive algorithms before describing how LACS leverages predictions to improve average-case performance without losing competitive guarantees.

Competitive baseline algorithms. Here we present two adaptations of the RORO framework detailed in Section 3.1, denoted by $\mathtt{RORO}_{\mathrm{cmax}}$ and $\mathtt{RORO}_{\mathrm{cmin}}$. Since OCSU introduces job length uncertainty, each of these adaptations considers an opposing extreme case for the job length. We describe each variant in turn below.

$\texttt{RORO}_{\text{cmax}}$: RORO *assuming maximum job lengths.* $\texttt{RORO}_{\text{cmax}}$ takes an optimistic approach by assuming every job has the maximum length $c_{\max}$. This aims to prepare for potentially long jobs by gathering enough resources. $\texttt{RORO}_{\text{cmax}}$'s assumption of worst-case job sizes makes it less conservative in terms of carbon intensities where it is willing to run the job. This gives it the flexibility to prepare for scenarios where long jobs (e.g., with length $c_{\max}$) do arise, although it risks "overspending" for shorter jobs. We see the impact of this assumption in $\texttt{RORO}_{\text{cmax}}$'s threshold function Equation 6. Compared to alternatives like RORO, which knows the exact job length, $\texttt{RORO}_{\text{cmax}}$'s threshold reduces at a slower exponential rate as job progress increases – intuitively, this is because $\texttt{RORO}_{\text{cmax}}$ plans for a *longer* job, which scales up the threshold function along the axis of w. Though this strategy may run the job when carbon intensities are "too high," particularly for jobs that are much shorter than $c_{\max}$, it can be advantageous when job lengths do approach the maximum. In such situations, $\texttt{RORO}_{\text{cmax}}$ may result in a more favorable outcome, avoiding the last-minute compulsory execution.

Definition 3.1. *The threshold function ϕ_1 used by $\texttt{RORO}_{cmax}$ for any progress $w \in [0, c_{max}]$ is defined as:*

$$\phi_1(w) = U - \beta + \left(\frac{U}{\alpha} - U + 2\beta\right) \exp\left(\frac{w}{c_{max}\alpha}\right), \tag{6}$$

where α is defined in Equation 5.

This approach captures one of two extreme cases that inform our algorithm design. Next, we will "flip" these assumptions to capture the other extreme case in the $\texttt{RORO}_{\text{cmin}}$ algorithm.

$\texttt{RORO}_{\text{cmin}}$: RORO *assuming minimum job lengths.* $\texttt{RORO}_{\text{cmin}}$ takes a pessimistic approach by assuming each job is as short as $c_{\min}$. This approach is efficient for handling shorter jobs since $\texttt{RORO}_{\text{cmin}}$ is more conservative in choosing which carbon intensities are good enough to run the job. The threshold function decreases faster than $\texttt{RORO}_{\text{cmax}}$, which assumes the maximum job length.

However, when $\texttt{RORO}_{\text{cmin}}$ encounters a longer job, its conservative nature can become a hindrance. $\texttt{RORO}_{\text{cmin}}$ is, by design, reluctant to allocate resources liberally due to its lower threshold, potentially missing the chance to make significant progress on lengthy jobs early on. This may necessitate costly compulsory execution at the end of the time period.

To mitigate this, we scale the threshold by the ratio $c_{\max}/c_{\min}$. This adjustment still allows the threshold to remain more cautious than $\texttt{RORO}_{\text{cmax}}$, but avoids the worst-case scenario for jobs that may turn out to be longer than $c_{\min}$. This remains economical for the assumed short jobs while also being flexible enough to accommodate the resource needs of unexpectedly longer jobs without resorting to compulsory executions at the end of the time period.

Definition 3.2. *The threshold function ϕ_2 used by $\texttt{RORO}_{cmin}$ for any progress $w \in [0, c_{max}]$ is defined as:*

$$\phi_2(w) = U - \beta + \left(\frac{U}{\alpha'} - U + 2\beta\right) \exp\left(\frac{w}{c_{max}\alpha'}\right), \tag{7}$$

where α' is defined as follows:

$$\alpha' = \left[\frac{c_{max}}{c_{min}} W\left[\frac{c_{min}}{c_{max}}\left(\frac{2\beta}{U} + \frac{L}{U} - 1\right)\exp\left(\frac{c_{min}}{c_{max}}\left(\frac{2\beta}{U} - 1\right)\right)\right] - \frac{2\beta}{U} + 1\right]^{-1}. \tag{8}$$

These two approaches comprise our worst-case algorithm design. While the competitive bounds of each algorithm differ, the empirical performance of each intuitively depends on the actual observed job lengths. In practice, we may often have a relatively accurate prediction about a given job's length. In the next section, we consider how this type of job length prediction can be incorporated into an algorithm design without losing worst-case guarantees.

Learning-augmented algorithm design. Here we formalize our learning-augmented algorithm, referred to as LACS and outlined in Algorithm 2. This algorithm integrates insights from two robust algorithms, $\texttt{RORO}_{\text{cmax}}$ and $\texttt{RORO}_{\text{cmin}}$, alongside predictions from an algorithm named $\texttt{RORO}_{\text{pred}}$. $\texttt{RORO}_{\text{pred}}$ is essentially a RORO algorithm that uses the predicted job length $\hat{c}$ rather than the actual job length c. Although $\texttt{RORO}_{\text{pred}}$ operates on predictions, it guarantees that the job is completed before the deadline (Equation 2) by beginning a compulsory execution when the remaining time steps are enough to complete a job with length $c_{\max}$.

The combination of $\texttt{RORO}_{\text{pred}}$ with competitive baselines $\texttt{RORO}_{\text{cmax}}$ and $\texttt{RORO}_{\text{cmin}}$ is designed to enhance average performance. Since $\texttt{RORO}_{\text{cmax}}$ is tailored for longer jobs and $\texttt{RORO}_{\text{cmin}}$ is more effective for shorter ones, we introduce an intermediate algorithm called $\texttt{RORO}_{\text{robust}}$ which leverages strengths of both competitive baselines. Let $k \in [0, 1]$ denote a decision factor, which dictates the proportion of the solution to derive from $\texttt{RORO}_{\text{cmax}}$ ($\{x_{1t}\}_{\forall t \in [T]}$) or $\texttt{RORO}_{\text{cmin}}$ ($\{x_{2t}\}_{\forall t \in [T]}$). Then $\texttt{RORO}_{\text{robust}}$ constructs a solution ($\{\tilde{x}_t\}_{\forall t \in [T]}$), where each $\tilde{x}_t$ is defined as $\tilde{x}_t = kx_{1t} + (1-k)x_{2t}, \forall t \in [T]$.

By unifying the two competitive baseline algorithms, we simplify the expression of LACS, which integrates these competitive decisions $\tilde{x}_t$ with the decisions of $\texttt{RORO}_{\text{pred}}$ as follows: We set an *augmentation factor* $\lambda \in [0, 1]$, which determines the influence of each algorithm on the final decision (λ from $\texttt{RORO}_{\text{pred}}$, $(1-\lambda)$ from $\texttt{RORO}_{\text{robust}}$). Intuitively, a larger value of λ implies that LACS is closer to the prediction. The result is a solution that benefits from the predictive strength of $\texttt{RORO}_{\text{pred}}$ while maintaining the robustness provided by the combined solutions of $\texttt{RORO}_{\text{cmax}}$ and $\texttt{RORO}_{\text{cmin}}$.

In the following, we formalize our instantiation of LACS (and $\texttt{RORO}_{\text{robust}}$ as a subroutine) for OCSU, which includes definitions of k, λ, and the parameters of ϵ and γ that they depend on.

Definition 3.3. *Let $\epsilon \in [0, |\alpha_1 - \alpha_2|]$, where α_1 and α_2 are the robust competitive ratios defined in Equation 9 and Theorem 4.2.*

We set $k = 1 - \frac{\epsilon}{\alpha_1 - \alpha_2}$ (which is bounded in $[0, 1]$) to form the solution ($\{\tilde{x}_t\}_{\forall t \in [T]}$) obtained by $\texttt{RORO}_{robust}$.

Let $\gamma \in [0, \alpha_1 - \texttt{sign}(\alpha_1 - \alpha_2)\epsilon - \alpha]$, where α is the robust competitive ratio defined in Equation 5, and $\texttt{sign}(x)$ is the sign function.

LACS *sets a augmentation factor of $\lambda = 1 - \frac{\gamma}{\alpha_1 - \texttt{sign}(\alpha_1 - \alpha_2)\epsilon - \alpha}$ which is bounded in $[0, 1]$.*

In the next section, we provide the consistency and robustness bounds for LACS. Intuitively, the primary desiderata for LACS is to be able to nearly match the performance of an online algorithm which knows the exact job length (e.g., RORO) when the job length predictions are correct, while preserving worst-case performance bounds in line with that of $\texttt{RORO}_{\text{cmax}}$ and $\texttt{RORO}_{\text{cmin}}$.

Algorithm 2 LACS: A *learning-augmented* algorithm for OCSU

input: The predicted solution $\{\hat{x}_t\}_{\forall t\in[T]}$ given by $\text{RORO}_{\text{pred}}$, competitive solutions $\{x_{1t}\}_{\forall t\in[T]}$ and $\{x_{2t}\}_{\forall t\in[T]}$ given by $\text{RORO}_{\text{cmax}}$ and $\text{RORO}_{\text{cmin}}$, decision factor k, augmentation factor λ.
while cost function $g_t(\cdot)$ is revealed and $w^{(t-1)} < c$ **do**
  obtain robust decisions x_{1t} and x_{2t};
  $\tilde{x}_t = kx_{1t} + (1-k)x_{2t}$;
  obtain prediction decision $\hat{x}_t$;
  set the online decision as $x_t = \lambda\hat{x}_t + (1-\lambda)\tilde{x}_t$;
  update the progress $w^{(t)} = w^{(t-1)} + x_t$;

4 THEORETICAL RESULTS

In this section, we state our main theoretical results. We start with the competitive results for the competitive baseline $\text{RORO}_{\text{cmax}}$ and $\text{RORO}_{\text{cmin}}$ algorithms before stating the consistency and robustness results for LACS. We discuss the results and their significance here, while deferring their full proofs to Appendix A.

Competitive analysis for $\text{RORO}_{\text{cmax}}$. Recall that $\text{RORO}_{\text{cmax}}$ assumes each job has the maximum length $c_{\max}$. In the following theorem, we state the competitive result for $\text{RORO}_{\text{cmax}}$ and discuss its significance. The full proof of Theorem 4.1 is in Section A.1.

Theorem 4.1. RORO_{cmax} *for* OCSU *is* α_1*-competitive when the threshold function is given by* $\phi_1(w)$ *from Definition 3.1.*

$$\alpha_1 = \frac{U}{\alpha L} + \frac{2\beta}{L}. \tag{9}$$

Intuitively, compared to the competitive bound α shown for OCS when job lengths are exactly known, α_1 is worse. This captures an edge case where the actual job length is, e.g., $c_{\min}$, while $\text{RORO}_{\text{cmax}}$'s design assumes the job has length $c_{\max}$. As we discuss in the full proof, this occurs because $\text{RORO}_{\text{cmax}}$'s scaled threshold design allows it to complete a job with length $c_{\min}$ by using the *worst* c_{min} *fraction* of the threshold, while RORO (where the job length is known) uses the entire domain of the monotonically decreasing threshold function to complete the job.

Competitive analysis for $\text{RORO}_{\text{cmin}}$. Contrary to the assumption of $\text{RORO}_{\text{cmax}}$, the $\text{RORO}_{\text{cmin}}$ algorithm is derived to prepare for a job with length $c_{\min}$, while acknowledging that the actual job length may be $c_{\max}$. In the following theorem, we state the competitive result for $\text{RORO}_{\text{cmin}}$ and discuss both its significance and relation to the existing RORO algorithm with known job lengths.

Theorem 4.2. RORO_{cmin} *for* OCSU *is* α'*-competitive when the threshold function is given by* $\phi_2(w)$ *from Definition 3.2. We henceforth use* $\alpha_2 = \alpha'$ *to denote the competitive ratio of* RORO_{cmin}.

As we show in the full proof, in Section A.2, $\alpha_2 = \alpha' \geq \alpha$, further implying that $\phi_2(w) \leq \phi_1(w)$ for any $w \in [0, c_{\max}]$. This supports the notion that $\text{RORO}_{\text{cmin}}$ is indeed more conservative than $\text{RORO}_{\text{cmax}}$ in terms of the carbon intensities for which it is willing to run the job. We note that the functions $\phi_1(w)$ and $\phi_2(w)$ are equivalent when $c_{\min} = c_{\max}$, indicating that both algorithms make the same decisions when all jobs have the same length.

Consistency and robustness of LACS. Recall that LACS (summarized in Algorithm 2) is our learning-augmented algorithm that plays a convex combination of the solutions obtained by $\text{RORO}_{\text{robust}}$ and $\text{RORO}_{\text{pred}}$. Letting $\alpha^{\max}_{\text{RORO}_{\text{pred}}}$ denote the worst-case competitive ratio of $\text{RORO}_{\text{pred}}$ (i.e., when the job length predictions are maximally incorrect), we obtain the following consistency and robustness bounds for LACS for any value of $\epsilon \in [0, |\alpha_1 - \alpha_2|]$ and any value of $\gamma \in [0, \alpha_1 - \texttt{sign}(\alpha_1 - \alpha_2)\epsilon - \alpha]$:

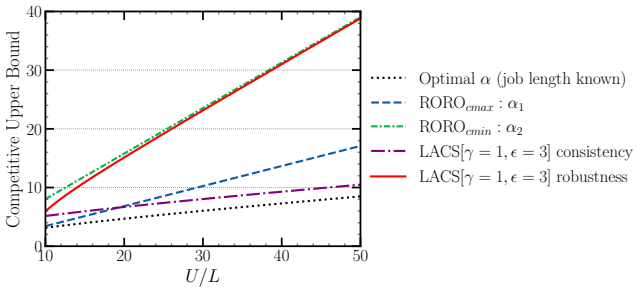


Figure 1: Competitive upper bounds for different algorithms (RORO with the full knowledge of job length, $\text{RORO}_{\text{cmin}}$, $\text{RORO}_{\text{cmax}}$, and $\text{LACS}[\gamma = 1, \epsilon = 3]$)

Theorem 4.3. *Given parameters* ϵ *and* γ, LACS *is* $(\alpha+\gamma)$*-consistent and* $\left[\left(1 - \frac{\gamma}{\alpha_1 - \texttt{sign}(\alpha_1-\alpha_2)\epsilon - \alpha}\right)\alpha^{\max}_{\text{RORO}_{pred}} + \left(\frac{\gamma(\alpha_1 - \texttt{sign}(\alpha_1-\alpha_2)\epsilon)}{\alpha_1 - \texttt{sign}(\alpha_1-\alpha_2)\epsilon - \alpha}\right)\right]$*-robust.*

This result, proven fully in Section A.4, implies that LACS can achieve a competitive ratio of α when the job length prediction is correct. Since α is the best achievable competitive ratio for the original OCS problem, LACS with accurate predictions of the job length achieves the optimal consistency bound as $\gamma \to 0$. Furthermore, the robustness bound implies that the worst-case competitive ratio when predictions are *incorrect* remains bounded by a combination of α_1 and α_2; this is intuitive because the competitive bounds of $\text{RORO}_{\text{cmax}}$ and $\text{RORO}_{\text{cmin}}$ (respectively) are the worst-case results for algorithms which expect one extreme job length and must deal with the other extreme job length.

In Figure 1, we plot the numerical values of α, α_1, α_2, and the consistency-robustness results of LACS with $\gamma = 1, \epsilon = 3$ for several different values of U/L. β is fixed to $U/10$, and $c_{\max}/c_{\min} = 4$. Note that α (the best competitive ratio for standard OCS) grows sublinearly in U/L, while the competitive bounds of $\text{RORO}_{\text{cmax}}$ (α_1) and $\text{RORO}_{\text{cmin}}$ (α_2) grow linearly in U/L. This reflects the inherent challenges of OCSU and the impact of uncertain job lengths on the achievable competitive ratios. Notably, for this setting of ϵ and γ, LACS is able to nearly match the optimal α for OCS when the predictions are correct (consistency), and is strictly upper bounded by α_2 when the predictions are adversarially incorrect (robustness), meaning that it achieves the *best of both worlds*.

5 EXPERIMENTAL RESULTS

In this section, we experimentally evaluate the performance of LACS in reducing the carbon footprint of scalable computing workloads.

5.1 Experimental Setup

We take a job-centric approach where a carbon-aware scheduler independently allocates (i.e., scales) computing resources to each job to reduce the total carbon footprint of its execution while respecting per-job deadlines. We next detail our experimental setup.

Carbon intensity trace. We use carbon intensity data from Electricity Maps [32] for California ISO (CAISO). The carbon trace

Table 2: Inverse of scaling profiles as a mapping between the completed part of the job (x) and amount of resources allocated (s).

Profile	Equation	Profile	Equation
P1	$s = x$	P4	$s = 0.5x^2 + x$
P2	$s = 0.15x^2 + x$	P5	$s = 0.75x^2 + x$
P3	$s = 0.25x^2 + x$	P6	$s = x^2 + x$

Table 3: Summary of characteristics for the baseline algorithms and two variants of the proposed algorithm

Algorithm	Carbon-Aware	Switching-Aware	Job Length Input
RORO [25]	Yes	Yes	Actual
OWT_{pred}	Yes	No	Prediction
Single Threshold	Yes	No	N/A
Carbon Agnostic	No	No	N/A
CarbonScaler [20]	Yes	No	Prediction
LACS (this paper)	Yes	Yes	Prediction
D-LACS[3](this paper)	Yes	Yes	Prediction

provides carbon intensity measured in grams of CO2 equivalent per kilowatt-hour (gCO2eq/kWh) at an hourly granularity and spans 2020 to 2023. We picked 1314 time slots as the job arrival times (once every 20 hrs[2]) to assess the performance across the whole trace duration. To evaluate algorithms that require carbon intensity (CI) forecasts, we introduce a uniformly random error to carbon intensity data to account for forecast errors, denoted as CI_{err}, where err is the mean of added percentage error.

Job characteristics. Each job arrives independently with a job length c uniformly sampled within the range $[c_{\min}, c_{\max}]$. To evaluate the impact of job length prediction error, we model a predictor that yields a job length estimate within the range $[c - p \times c, c + p \times c]$, where p is the percentage error in job length predictions. We also assume that all jobs have a deadline (T) of 24 hrs and incur a fixed symmetric maximum switching overhead of $\beta.h(r)$ (e.g., for checkpoint and resume) when scaling from zero to the maximum resources r, where $h(\cdot)$ is the scaling profile.

Resource scaling profiles. Carbon savings highly depend on the scalability of jobs [20], where more scalable jobs can yield higher savings as they provide higher marginal throughput for each added resource. Table 2 depicts the mapping functions $s = h_t^{-1}(x)$ (the inverse of the scaling profiles), where x represents the completed part of the job (i.e., progress made) and s represents the amount of resource, e.g., servers, to obtain the given progress. Some of the utilized profiles represent common scalability profiles of real-world batch jobs. For instance, P1 refers to embarrassingly parallel applications such as BLAST [41], while P2 and P4 are a fitted version of the machine learning training workloads for ResNet18 and MobileNetV2, respectively, described in [20]. In contrast, P3, P4, and P6 are synthesized profiles to represent moderate and non-scalable applications.

Parameter settings. We evaluate LACS across a wide range of experimental scenarios that impact its performance, including a range of maximum job lengths ($c_{\max}$), varying coefficients for switching emissions (β), errors in job length predictions, and a range of learning augmentation factors (λ). We set the value of the decision factor (k) in Algorithm 2 to 0.5, so both robust algorithms receive equal consideration. To impose practical constraints, we evaluate LACS across a range of carbon intensity forecast errors (CI_{err}) and resource constraints (r) since available resources are constrained.

Baseline algorithms. We evaluate the two variants of our proposed algorithm, LACS and D-LACS, against multiple state-of-the-art algorithms, summarized in Table 3 and detailed below.

(1) **RORO:** An online conversion with switching costs algorithm with knowledge of job length [25], described in Section 3.1. This *online algorithm* knows the accurate job length, so it serves as an upper bound for LACS that has no access to the exact job length a priori.
(2) **OWT_{pred}:** Threshold-based one-way trading with job length predictions. This algorithm adapts a threshold-based one-way trading algorithm [47] that assumes perfect job length information. Our adaptation, OWT_{pred}, uses inaccurate job length predictions to decide the amount of job x_t to be scheduled at time t based on a threshold function Φ. It accounts for the carbon emissions of execution but ignores any switching emissions. When $\beta = 0$, $\text{RORO}_{\text{pred}}$ (described in Section 3.2) reduces to OWT_{pred}.
(3) **Single Threshold:** The algorithm utilizes a static threshold set at $\sqrt{UL}$, a value initially introduced in [16]. Adapted for OCSU, Single Threshold operates by running the job with the maximum resource available at each time step t, but only if the execution emissions are lower than $\sqrt{UL}$. This algorithm also does not consider switching emissions.
(4) **Carbon Agnostic:** This algorithm presents a greedy approach that executes each job with maximum available resources upon submission. As the scheduler does not know the job length, executing the job with maximum resources reduces execution time and ensures completion before the deadline, if feasible.
(5) **CarbonScaler:** The CarbonScaler [20] algorithm utilizes its knowledge of the job length, carbon intensity, and deadline to construct a carbon-aware resource scaling. To accommodate job length prediction inaccuracies and potential errors in carbon intensity forecasts, we feed the predicted job length and the erroneous carbon intensity forecasts to the algorithm.

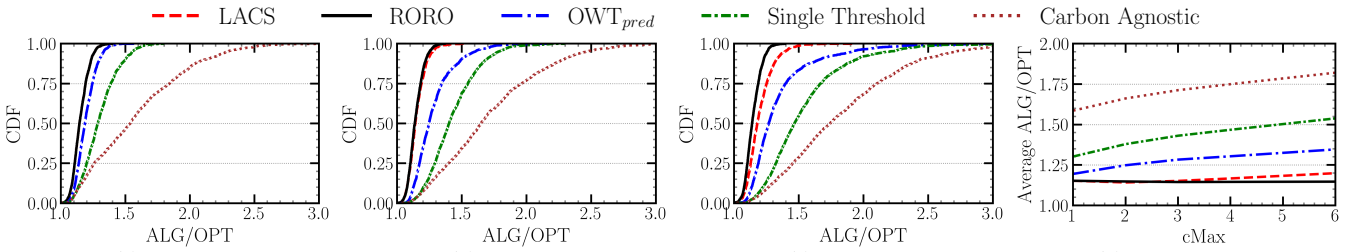
It must be noted that all algorithms, except RORO, start the compulsory execution at the time slot $T - (c_{\max} - w^{(t)})/d_t$, where T is the deadline, and $w^{(t)}$ is the progress at time t to ensure that the job completes before the deadline[4].

Evaluation metric. We compare the performance of an algorithm against an offline optimal resource scaling schedule computed using a numerical solver [52], which allows us to report the empirical competitive ratio represented as ALG/OPT (lower value is better). We also report the reduction in carbon footprint with respect to the Carbon Agnostic execution of the job that uses maximum resources and aims to finish the job as soon as possible.

5.2 Effect of Maximum Job Length

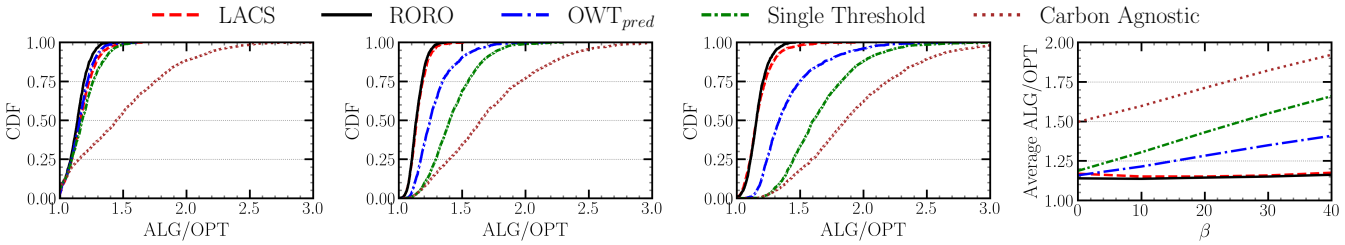
The ratio between the maximum and minimum job lengths ($c_{\max}$, $c_{\min}$) dictates the characteristics of jobs that may arrive at the scheduler; a higher ratio means the job lengths can be more diverse. For simplicity, we set $c_{\min} = 1$ and analyze the effect of $c_{\max}$. Figure 2 shows the performance of various algorithms under different $c_{\max}$ values, where lower ALG/OPT is better. We evaluate the proposed algorithm against RORO, OWT_{pred}, Single Threshold, and Carbon

[2]We purposefully avoid an arrival every 24 hrs to avoid diurnal patterns.
[3]D-LACS is the modified version of LACS, which considers discrete resource allocation based on the solution given by LACS.
[4]RORO starts the compulsory execution based the actual job length c



(a) c_{max} = 1 (b) c_{max} = 3 (c) c_{max} = 6 (d) Effect of c_{max}

Figure 2: (a), (b), and (c) report cumulative distribution functions (CDFs) of empirical competitive ratios for evaluated algorithms under different c_{max} values, where c_{min} =1. (d) Shows the effect of c_{max} on empirical competitive ratios. The scaling profile is P1, β = 20, job prediction error is 20%, and λ = 0.5. A CDF curve towards the top left corner indicates better performance.



(a) β = 0 (b) β = 20 (c) β = 40 (d) Effect of β

Figure 3: (a), (b), and (c) report cumulative distribution functions (CDFs) of empirical competitive ratios for selected algorithms under different β values. (d) Shows the effect of β on empirical competitive ratios. The scaling profile is P1, c_{max} = 3, job prediction error is 20%, and λ=0.5. A CDF curve towards the top left corner indicates better performance.

Agnostic by showing their performance compared to the offline optimal algorithm. We omit CarbonScaler in this section as it requires discrete resource allocation; we will consider it when we enforce discrete assignment. We assume scaling profile P1, i.e., the job is embarrassingly parallel, switching emission coefficient β = 20, job prediction error of 20%, and augmentation factor λ = 0.5.

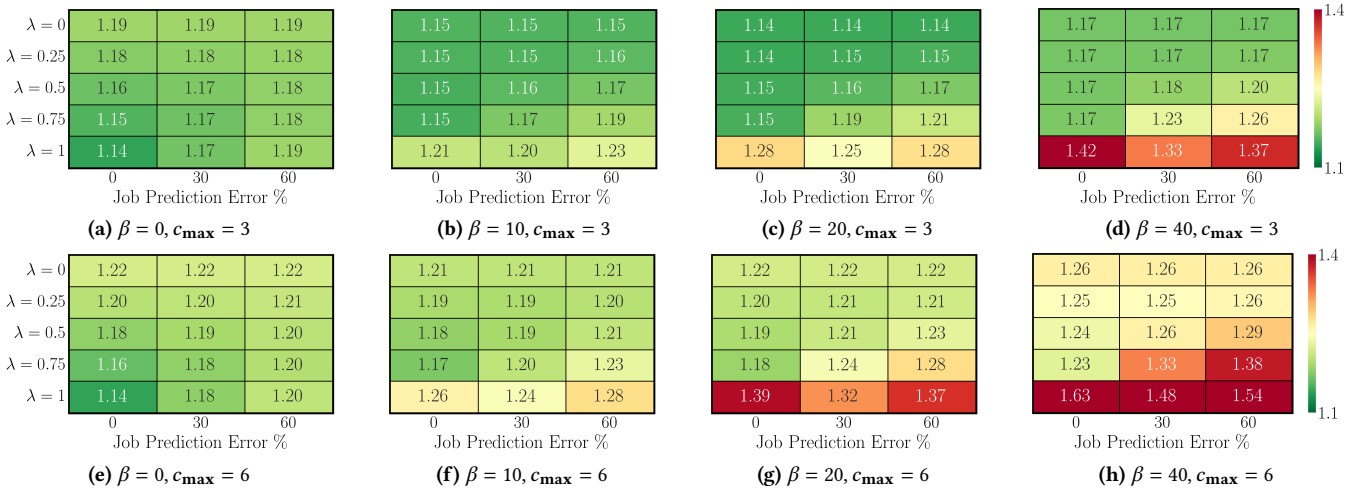
Figure 2a shows the special case where $c_{max} = c_{min}$, i.e., all jobs are the same length and known to LACS, RORO, OWT$_{pred}$. As expected, LACS and RORO are identical while OWT$_{pred}$ is slightly behind as it does not consider switching emissions. Nonetheless, these algorithms achieve higher performance than Single Threshold and Carbon Agnostic. Figure 2b and Figure 2c show realistic cases where the c_{max} is 3 and 6, respectively. As shown, LACS is the closest to RORO in all settings. Its performance is within 15.1 and 19.8% on average from the offline optimal and only 0.6% and 4.6% from RORO, when c_{max} is 3 and 6, respectively. Figure 2d summarizes the average competitive ratio across different c_{max} values; increasing the c_{max} decreases the performance of all techniques, as a higher c_{max} yields more uncertainty on the actual job length and exacerbates switching emissions as we do not employ resource constraints. Nonetheless, the results indicate the superiority of LACS across different c_{max} values, where LACS performs within 16.1% of the offline optimal and only 1.3% away from RORO, resulting in a 31% reduction in carbon emissions compared to the Carbon Agnostic policy.

5.3 Effect of Switching Emissions

Dynamic resource adjustment leads to wasted time and energy, which result in extra emissions. For example, when applications employ suspend-resume scheduling, the switching emissions are due to the incurred energy required to checkpoint or restore the state before being able to resume processing. Since these switching emissions differ across systems and applications, e.g., due to memory size [40], we evaluate the effect of the switching emissions coefficient β on performance, where higher β hinders the policy's ability to adapt to variations in carbon intensity. Figure 3 shows the performance of the algorithms against different values of β. We fix maximum job length (c_{max} = 3) while keeping the other parameters the same as Figure 2. We assigned β as 0, 20, and 40 gCO2eq, representing 0.0, 7.3, and 14.6% of California's average hourly carbon intensity (273 gCO2eq/kWh), respectively. For completeness, we added the case of β = 0 where resource scaling does not incur any overheads. Figure 3a shows the case where switching is cost-free. We observe that OWT$_{pred}$ outperforms LACS, because LACS must also incorporate the robust decisions from RORO$_{robust}$ (note that since β = 0, OWT$_{pred}$ is equivalent to RORO$_{pred}$). Figure 3b and Figure 3c explore realistic cases with non-zero switching emissions, where LACS nearly matches RORO. It lags behind on average and in the worst case by 0.6 and 10%, respectively, when β = 20 and by 1.1 and 32.3%, respectively, when β = 40. Finally, Figure 3d summarizes the average performance of all the algorithms, highlighting the effectiveness of our proposed approach. As expected, higher β leads to higher ALG/OPT for the algorithms that do not incorporate switching emissions. However, increasing β does not affect the relative performance of the algorithms; LACS achieves average performance within 1.2% of RORO and 16% of the offline optimal, which constitutes 32% carbon savings compared to Carbon Agnostic.

5.4 Effect of Job Length Prediction Error

As explained in Section 3.2, LACS augments robust algorithms with a prediction-based approach that leverages job length predictions. However, since typical batch job predictors are highly erroneous, we use an augmentation factor λ that controls how much this job length prediction influences the final solution. Figure 4 shows the



(a) $\beta = 0, c_{\max} = 3$ **(b)** $\beta = 10, c_{\max} = 3$ **(c)** $\beta = 20, c_{\max} = 3$ **(d)** $\beta = 40, c_{\max} = 3$

(e) $\beta = 0, c_{\max} = 6$ **(f)** $\beta = 10, c_{\max} = 6$ **(g)** $\beta = 20, c_{\max} = 6$ **(h)** $\beta = 40, c_{\max} = 6$

Figure 4: Average competitive ratios for LACS under different job prediction errors, augmentation factors λ, switching emissions β and $c_{\max}$ values for scaling profile P1.

effect of various prediction accuracies, defined by prediction error %, and augmentation factors λ on the empirical competitiveness of LACS under different maximum job lengths $c_{\max}$ and switching emissions coefficients β. The scaling profile is P1.

The results show that as the prediction error increases, LACS should employ a lower augmentation factor for the predictions. For instance, at a job prediction error of 60%, a mid-range augmentation factor (between 0.25 and 0.75) leads to better results than either extreme. Interestingly, the results show that higher prediction accuracy and augmentation factors do not always guarantee the highest performance. For instance, for 0% error in job length predictions, fully augmenting with predictions does not yield the best ALG/OPT. For example, when $\beta = 10$, $c_{\max} = 3$ (Figure 4b) and $\beta = 40$, $c_{\max} = 6$ (Figure 4h) setting $\lambda = 0.75$ outperforms fully augmenting with predictions ($\lambda = 1$) by 21% and 33%, respectively.

This counter-intuitive result manifests because, during compulsory execution, $\text{RORO}_{\text{pred}}$ must run with the maximum available resources, regardless of the job length prediction accuracy. This can lead to increased switching emissions compared to scenarios where robust algorithms contribute more to the decision-making in LACS, potentially reducing the time spent in the compulsory execution. This may happen since the robust algorithm caters to the upper bound and sets a less conservative threshold. Higher values of β and $c_{\max}$ can intensify this phenomenon, as more switching emissions are incurred when scheduling with the maximum available resources in the compulsory execution zone. In summary, LACS balances the decisions from the robust algorithm and the job length predictor by using a moderate augmentation factor λ; the results indicate that aside from different values of $c_{\max}$, β, and job length prediction errors, an augmentation factor $\lambda = 0.5$ can perform within 4% of RORO and within 18% of the offline optimal, which translates to 20.8% carbon savings over Carbon Agnostic.

5.5 Real-world Considerations

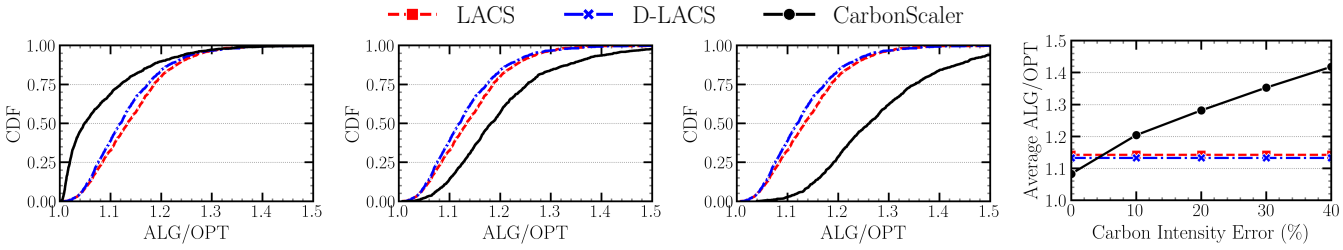
In the previous experiments, we assumed that resource allocation is continuous and resources can be acquired in any quantity. In practice, however, resources such as cores or servers must be allocated in discrete quantities and have physical and performance constraints. To accommodate such requirements, we discretize the scaling profile into fixed-size units, which map to different amounts of the job based on the scalability of the profile. Then, to map the scheduling decisions to discrete allocations, we round the decisions of LACS to the nearest discrete quantity, a policy we denote Discrete-LACS (D-LACS). To create this discrete allocation, we dissect each resource unit into 8 equally sized segments. Our evaluation shows that the discretization process has a negligible impact on our results, as observed across all the results in Figure 5 and Figure 6.

Additionally, the maximum resources available to a job may be constrained for various reasons, including resource contention and cost considerations. To accommodate such constraints and analyze their effect, we bound the resources allocated to the job in a time slot to a maximum number r. We assume that the total number of resource units is 32, where resource r takes values of 1/8, 1/4, 1/2, and 1, denoting 4, 8, 16, and 32 resource units. This ensures compatibility with the employed resource discretization. In what follows, we evaluate the performance of these considerations and compare our proposed method to CarbonScaler [20] that assumes both discrete resources and rate limits but relies on carbon intensity forecasts to make scheduling decisions.

Impact of carbon intensity forecast error. The requirement of fairly accurate carbon intensity forecasts significantly limits the practical deployment of CarbonScaler [20]. We evaluate the effect of CI_{err} on the performance of CarbonScaler and compare it with LACS that does not require CI forecasts. In the experimental setup for Figure 5, we choose profile P2[5], impose a resource constraint (r=1/4), consider job length prediction error of 30%, while keeping the rest of the setup the same as Figure 3. We evaluate the performance of LACS, D-LACS[6], and CarbonScaler under different CI_{err} values.

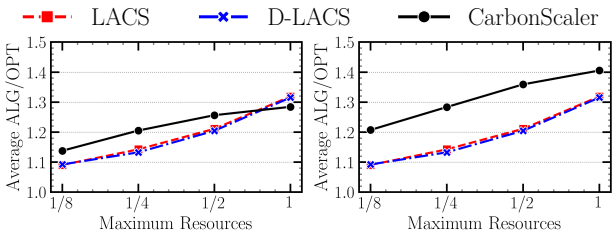
Figure 5a shows the baseline case without any forecast errors, i.e., $\text{CI}_{\text{err}} = 0$, and shows that although CarbonScaler outperforms

[5]We do not use CarbonScaler with P1 as it will scale to the maximum resources during expected low carbon intensity slots, exacerbating the effect of CI_{err}.
[6]We note that the optimal policy does not mandate discrete assignments but considers the rate constraints, making it a lower bound for our solutions



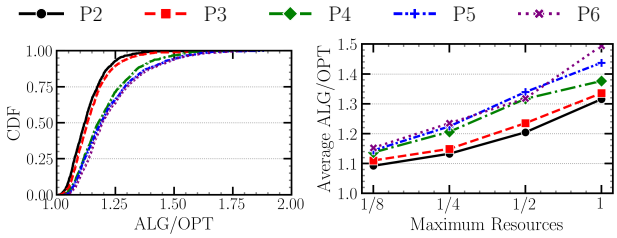
(a) CI_{err} = 0 (b) CI_{err} = 10% (c) CI_{err} = 20% (d) Effect of CI_{err}

Figure 5: (a), (b), and (c) report cumulative distribution functions (CDFs) of empirical competitive ratios for selected algorithms under different CI_{err} values. (d) shows the effect of CI_{err} on average ALG/OPT. We assume scaling profile is P2, c_{max} = 3, r = 1/4, β = 20, job prediction error is 30%, and λ = 0.5. A CDF curve towards the top left corner indicates better performance.



(a) CI_{err} = 10% (b) CI_{err} = 20%

Figure 6: Average competitive ratios across policies under different maximum resource constraints r and carbon intensity forecast error CI_{err}. We assume scaling profile is P2, c_{max} = 3, β = 20, job prediction error is 30% and λ = 0.5.



(a) Maximum resources is 1/4 (b) Effect of maximum resources

Figure 7: Effect of profiles on Average competitive ratio across scaling profiles from Table 2. We assume c_{max} = 3, r = 1/4, β = 20 job prediction error is 30%, CI_{err} = 10, and λ = 0.5.

D-LACS on average by 4.6%, D-LACS outperforms CarbonScaler by 4.8% in the worst-case. Figure 5b and Figure 5c show more realistic scenarios where the forecasts are erroneous and demonstrate the sensitivity of CarbonScaler to carbon intensity errors. At 10% error, which is equivalent to the average error rate of state-of-the-art carbon intensity forecasting models such as [31], the performance of CarbonScaler is strictly below D-LACS, where D-LACS outperforms CarbonScaler by 6.3 and 4.8% when CI_{err} = 10 and by 13.1 and 7.9% when CI_{err} = 20, on average and in the worst case, respectively. Figure 5d depicts the performance of CarbonScaler and shows the applicability of D-LACS in the real world with erroneous or unavailable carbon intensity forecasts.

Impact of resource constraints. Figure 6 shows the performance of LACS, D-LACS and CarbonScaler as a function of resource constraints. We set the scaling profile (P2), c_{max} = 3, switching emissions coefficient β = 20, the job prediction error to 30%, and augmentation factor as λ = 0.5. The results indicate that enforcing a lower resource constraint narrows the gap between algorithms and the offline optimal, where the average performance of D-LACS ranges from 9 to 31% of the offline optimal, rending 21 to 37% carbon savings compared to the Carbon Agnostic policy. This is reasonable as a lower resource constraint means a lower degree of freedom, forcing all approaches to run similarly. Nevertheless, D-LACS outperforms CarbonScaler in almost all cases. For example, when r = 1/8, D-LACS outperforms CarbonScaler by 3.6 and 10% when CI_{err} = 10 and CI_{err} = 20, respectively. The results are consistent with previous experiments, where higher CI_{err} yields a higher gap between D-LACS and CarbonScaler, reaching 13.3% when CI_{err} = 20 and r = 1/4. In the absence of resource constraints, the gap between CarbonScaler and D-LACS shrinks as CarbonScaler can fully scale up when encountering a good carbon intensity, possibly avoiding compulsory execution.

5.6 Effect of Scaling Profiles

Many factors, including the network speed, and the ratio between workload's communication and computation, determine an application's scalability. To evaluate the effect of scaling profiles, we test various profiles (see Table 2), which represent possible scaling profiles seen in the real world. Figure 7 evaluates the performance of D-LACS for these different profiles. We set c_{max} = 3, switching emissions coefficient β = 20, the job prediction error is 30%, the carbon intensity forecast error is set to CI_{err} = 10%, and augmentation factor λ = 0.5. Figure 7a shows the performance of D-LACS across different profiles; D-LACS is more effective at higher scalability, but shows comparable competitiveness across profiles. As resources increase, jobs with worse scaling profiles will be more conservative in their decisions, forcing them to run at inefficient scales during the compulsory execution. Figure 7b, depicts this behavior across scaling behavior and rates, showing that all scaling profiles experience less performance as the rate increases, which is consistent with earlier results. The results indicate that across scaling profiles D-LACS performs between 9 and 15% and 31 and 49% of the offline optimal for r = 1/8 and r = 1, respectively.

Table 4: Carbon-aware temporal shifting and scaling

Algorithm	Unknown Job Length	Forecast Not Required	Deadline	Switching Cost	Decision Space
WaitAwhile-Thr. [53]	Yes	Yes	No	No	Shift
WaitAwhile [53]	No	No	Yes	No	Shift
k-min search [28]	No	Yes	Yes	No	Shift
Double Threshold [26]	No	Yes	Yes	Yes	Shift
Wait&Scale [41]	Yes	No	No	No	Scale
CarbonScaler [20]	No	No	Yes	No	Scale
OWT [16, 47]	No	Yes	Yes	No	Scale
RORO [25]	No	Yes	Yes	Yes	Scale
This work	Yes	Yes	Yes	Yes	Scale

6 RELATED WORK

To bridge the gap between the availability of low-carbon energy and demand, carbon-aware schedulers utilize the inherent flexibility of workloads to select an appropriate time and location to execute the workloads [15, 19, 20, 23, 25, 26, 41, 42, 44, 50, 53, 57]. In this paper, we focus on a special case of carbon-aware temporal shifting of batch jobs, where schedulers decide on a scale factor at each time step ranging from 0 (i.e., suspending) to a user-defined max factor. In addition, the problem of carbon-aware scheduling has historically been closest to *online search problems* such as k-min search [27, 28], one-way trading [14, 16, 35, 45, 47], and online knapsack [5, 13, 33, 46, 54, 55, 58]. These have seen applications in e.g., cloud pricing [56], EV charging [8, 47], and stock trading [28], among others. Recently, two studies have explicitly extended online search ideas towards carbon-aware problems. In [26], the authors present online pause and resume, which extends the k-min search problem to incorporate switching costs. Similarly, the authors in [25] present the OCS problem discussed in this paper, which introduces a linear switching cost into the formulation of one-way trading. In contrast to all of the above literature, the OCSU problem we propose is the first online search-type setting where the job length (i.e., demand) is uncertain. We summarize state-of-the-art work in carbon-aware scheduling and applicable methods in online scheduling in Table 4, highlighting different assumptions regarding job length, dependency on carbon intensity forecasts, and compliance with deadlines.

7 CONCLUSION

This paper introduces LACS, a novel learning-augmented carbon-aware algorithm for resource scaling of computing workloads with uncertain job lengths. We analyzed the theoretical performance of LACS using the framework of robustness-consistency of competitive online algorithms. Further, we evaluated the empirical performance of LACS through extensive experimental, showing its superior performance as compared to an extensive set of baseline algorithms. LACS, to be best of our knowledge, is the first algorithm with both theoretical guarantees and promising practical performance for carbon-aware resource scaling with unknown job lengths.

ACKNOWLEDGMENTS

Mohammad Hajiesmaili acknowledges this work is supported by the U.S. National Science Foundation (NSF) under grant numbers CAREER-2045641, CPS-2136199, CNS-2106299, CNS-2102963, and NGSDI-2105494. Prashant Shenoy's research is supported by NSF grants 2213636, 2105494, 2021693, 2020888, DOE grant DE-EE0010143, and VMware.

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Department of Energy Computational Science Graduate Fellowship under Award Number DE-SC0024386.

DISCLAIMERS

REFERENCES

[1] Pradeep Ambati, Noman Bashir, David Irwin, and Prashant Shenoy. 2021. Good Things Come to Those Who Wait: Optimizing Job Waiting in the Cloud. In *Proceedings of the ACM Symposium on Cloud Computing* (Seattle, WA, USA) *(SoCC '21)*. Association for Computing Machinery, New York, NY, USA, 229–242. https://doi.org/10.1145/3472883.3487007

[2] Douglas J Arent, Peter Green, Zia Abdullah, Teresa Barnes, Sage Bauer, Andrey Bernstein, Derek Berry, Joe Berry, Tony Burrell, Birdie Carpenter, et al. 2022. Challenges and opportunities in decarbonizing the US energy system. *Renewable and Sustainable Energy Reviews* 169 (2022), 112939.

[3] Rohan Arora, Umamaheswari Devi, Tamar Eilam, Aanchal Goyal, Chandra Narayanaswami, and Pritish Parida. 2023. Towards Carbon Footprint Management in Hybrid Multicloud. In *Proceedings of the 2nd Workshop on Sustainable Computer Systems*. 1–7.

[4] Noman Bashir, David Irwin, Prashant Shenoy, and Abel Souza. 2023. Sustainable Computing – Without the Hot Air. *ACM SIGENERGY Energy Informatics Review* 3, 3 (2023), 47–52.

[5] Hans-Joachim Böckenhauer, Dennis Komm, Richard Královič, and Peter Rossmanith. 2014. The online knapsack problem: Advice and randomization. *Theoretical Computer Science* 527 (2014), 61–72. https://doi.org/10.1016/j.tcs.2014.01.027

[6] Allan Borodin and Ran El-Yaniv. 2005. *Online computation and competitive analysis*. cambridge university press.

[7] Allan Borodin, Nathan Linial, and Michael E. Saks. 1992. An Optimal On-Line Algorithm for Metrical Task System. *J. ACM* 39, 4 (Oct 1992), 745–763. https://doi.org/10.1145/146585.146588

[8] Roozbeh Bostandoost, Bo Sun, Carlee Joe-Wong, and Mohammad Hajiesmaili. 2023. Near-Optimal Online Algorithms for Joint Pricing and Scheduling in EV Charging Networks. In *Proceedings of the 14th ACM International Conference on Future Energy Systems* (Orlando, FL, USA) *(e-Energy '23)*. Association for Computing Machinery, New York, NY, USA, 72–83. https://doi.org/10.1145/3575813.3576878

[9] Wesley J Cole, Danny Greer, Paul Denholm, A Will Frazier, Scott Machen, Trieu Mai, Nina Vincent, and Samuel F Baldwin. 2021. Quantifying the challenge of reaching a 100% renewable energy power system for the United States. *Joule* 5, 7 (2021), 1732–1748.

[10] European Commission. 2024. Carbon Border Adjustment Mechanism. https://taxation-customs.ec.europa.eu/carbon-border-adjustment-mechanism_en. Accessed January.

[11] Robert M Corless, Gaston H Gonnet, David EG Hare, David J Jeffrey, and Donald E Knuth. 1996. On the Lambert W function. *Advances in Computational mathematics* 5 (1996), 329–359.

[12] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. 2017. Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles* (Shanghai, China) *(SOSP '17)*. Association for Computing Machinery, New York, NY, USA, 153–167. https://doi.org/10.1145/3132747.3132772

[13] Marek Cygan, Łukasz Jeż, and Jiří Sgall. 2016. Online knapsack revisited. *Theory of Computing Systems* 58 (2016), 153–190.

[14] Peter Damaschke, Phuong Hoai Ha, and Philippas Tsigas. 2007. Online Search with Time-Varying Price Bounds. *Algorithmica* 55, 4 (Dec. 2007), 619–642. https://doi.org/10.1007/s00453-007-9156-9

[15] Jesse Dodge, Taylor Prewitt, Remi Tachet des Combes, Erika Odmark, Roy Schwartz, Emma Strubell, Alexandra Sasha Luccioni, Noah A Smith, Nicole DeCario, and Will Buchanan. 2022. Measuring the Carbon Intensity of AI in Cloud Instances. In *FAccT*. ACM, New York, NY, USA, 1877–1894.

[16] Ran El-Yaniv, Amos Fiat, Richard M. Karp, and G. Turpin. 2001. Optimal Search and One-Way Trading Online Algorithms. *Algorithmica* 30, 1 (May 2001), 101–139. https://doi.org/10.1007/s00453-001-0003-0

[17] Udit Gupta, Mariam Elgamal, Gage Hills, Gu-Yeon Wei, Hsien-Hsin S Lee, David Brooks, and Carole-Jean Wu. 2022. ACT: Designing Sustainable Computer Systems with an Architectural Carbon Modeling Tool. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*. ACM, New York, NY, USA, 784–799.

[18] Udit Gupta, Young Geun Kim, Sylvia Lee, Jordan Tse, Hsien-Hsin S. Lee, Gu-Yeon Wei, David Brooks, and Carole-Jean Wu. 2022. Chasing Carbon: The Elusive Environmental Footprint of Computing. *IEEE Micro* 42, 4 (jul 2022), 37–47. https://doi.org/10.1109/MM.2022.3163226

[19] Walid A. Hanafy, Roozbeh Bostandoost, Noman Bashir, David Irwin, Mohammad Hajiesmaili, and Prashant Shenoy. 2023. The War of the Efficiencies: Understanding the Tension between Carbon and Energy Optimization. In *Proceedings of the 2nd Workshop on Sustainable Computer Systems*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3604930.3605709

[20] Walid A. Hanafy, Qianlin Liang, Noman Bashir, David Irwin, and Prashant Shenoy. 2023. CarbonScaler: Leveraging Cloud Workload Elasticity for Optimizing Carbon-Efficiency. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 7, 3 (Dec 2023), 28 pages. arXiv:2302.08681 [cs.DC]

[21] Stephen P. Holland, Matthew J. Kotchen, Erin T. Mansur, and Andrew J. Yates. 2022. Why Marginal CO2 Emissions Are Not Decreasing for U.S. Electricity: Estimates and Implications for Climate Policy. *Proceedings of the National Academy of Sciences* 119, 8 (2022), e2116632119.

[22] Abdolhossein Hoorfar and Mehdi Hassani. 2008. Inequalities on the Lambert W function and hyperpower function. *Journal of Inequalities in Pure and Applied Mathematics* 9, 51 (Jan. 2008). Issue 2.

[23] Young Geun Kim, Udit Gupta, Andrew McCrabb, Yonglak Son, Valeria Bertacco, David Brooks, and Carole-Jean Wu. 2023. GreenScale: Carbon-Aware Systems for Edge Computing. *arXiv preprint arXiv:2304.00404* (2023).

[24] Michael Kuchnik, J. Park, C. Cranor, Elisabeth Moore, Nathan DeBardeleben, and George Amvrosiadis. 2019. *This is Why ML-driven Cluster Scheduling Remains Widely Impractical.* Technical Report CMU-PDL-19-103.

[25] Adam Lechowicz, Nicolas Christianson, Bo Sun, Noman Bashir, Mohammad Hajiesmaili, Adam Wierman, and Prashant Shenoy. 2024. Online Conversion with Switching Costs: Robust and Learning-augmented Algorithms. In *Proceedings of the 2024 SIGMETRICS/Performance Joint International Conference on Measurement and Modeling of Computer Systems* (Venice, Italy) *(SIGMETRICS / Performance '24)*. Association for Computing Machinery, New York, NY, USA.

[26] Adam Lechowicz, Nicolas Christianson, Jinhang Zuo, Noman Bashir, Mohammad Hajiesmaili, Adam Wierman, and Prashant Shenoy. 2023. The Online Pause and Resume Problem: Optimal Algorithms and An Application to Carbon-Aware Load Shifting. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 7, 3, Article 53 (Dec 2023), 36 pages. arXiv:2303.17551 [cs.DS]

[27] Russell Lee, Bo Sun, Mohammad Hajiesmaili, and John C. S. Lui. 2024. Online Search with Predictions: Pareto-optimal Algorithm and its Applications in Energy Markets. In *Proceedings of the 15th ACM International Conference on Future Energy Systems* (Singapore, Singapore) *(e-Energy '24)*. Association for Computing Machinery, New York, NY, USA.

[28] Julian Lorenz, Konstantinos Panagiotou, and Angelika Steger. 2008. Optimal Algorithms for k-Search with Application in Option Pricing. *Algorithmica* 55, 2 (Aug. 2008), 311–328. https://doi.org/10.1007/s00453-008-9217-8

[29] Alexandra Sasha Luccioni, Yacine Jernite, and Emma Strubell. 2023. Power Hungry Processing: Watts Driving the Cost of AI Deployment? arXiv:2311.16863 [cs.LG]

[30] Thodoris Lykouris and Sergei Vassilvtiskii. 2018. Competitive Caching with Machine Learned Advice. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 3296–3305. https://proceedings.mlr.press/v80/lykouris18a.html

[31] Diptyaroop Maji, Prashant Shenoy, and Ramesh K. Sitaraman. 2022. CarbonCast: Multi-Day Forecasting of Grid Carbon Intensity. In *Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation* (Boston, Massachusetts) *(BuildSys '22)*. Association for Computing Machinery, New York, NY, USA, 198–207. https://doi.org/10.1145/3563357.3564079

[32] Electricity Maps. 2023. Electricity Map. https://www.electricitymap.org/map.

[33] A. Marchetti-Spaccamela and C. Vercellis. 1995. Stochastic on-line knapsack problems. *Mathematical Programming* 68, 1-3 (Jan. 1995), 73–104. https://doi.org/10.1007/bf01585758

[34] Dragoslav S. Mitrinovic, Josip E. Pečarić, and A. M. Fink. 1991. *Inequalities Involving Functions and Their Integrals and Derivatives*. Vol. 53. Springer Science & Business Media.

[35] Esther Mohr, Iftikhar Ahmad, and Günter Schmidt. 2014. Online algorithms for conversion problems: a survey. *Surveys in Operations Research and Management Science* 19, 2 (2014), 87–104.

[36] Steven Gonzalez Monserrate. 2022. The Staggering Ecological Impacts of Computation and the Cloud. https://thereader.mitpress.mit.edu/the-staggering-ecological-impacts-of-computation-and-the-cloud/.

[37] National Science Foundation (NSF). 2022. Design for Environmental Sustainability in Computing (DESC). https://www.nsf.gov/pubs/2023/nsf23532/nsf23532.htm.

[38] Manish Purohit, Zoya Svitkina, and Ravi Kumar. 2018. Improving Online Algorithms via ML Predictions. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc.

[39] Krzysztof Rzadca, Pawel Findeisen, Jacek Swiderski, Przemyslaw Zych, Przemyslaw Broniek, Jarek Kusmierek, Pawel Nowak, Beata Strack, Piotr Witusowski, Steven Hand, and John Wilkes. 2020. Autopilot: workload autoscaling at Google. In *Proceedings of the Fifteenth European Conference on Computer Systems* (Heraklion, Greece) *(EuroSys '20)*. Association for Computing Machinery, New York, NY, USA, Article 16, 16 pages. https://doi.org/10.1145/3342195.3387524

[40] Prateek Sharma, Tian Guo, Xin He, David Irwin, and Prashant Shenoy. 2016. Flint: batch-interactive data-intensive processing on transient servers. In *Proceedings of the Eleventh European Conference on Computer Systems* (London, United Kingdom) *(EuroSys '16)*. Association for Computing Machinery, New York, NY, USA, Article 6, 15 pages. https://doi.org/10.1145/2901318.2901319

[41] Abel Souza, Noman Bashir, Jorge Murillo, Walid Hanafy, Qianlin Liang, David Irwin, and Prashant Shenoy. 2023. Ecovisor: A Virtual Energy System for Carbon-Efficient Applications. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2* (Vancouver, BC, Canada) *(ASPLOS 2023)*. Association for Computing Machinery, New York, NY, USA, 252–265. https://doi.org/10.1145/3575693.3575709

[42] Abel Souza, Shruti Jasoria, Basundhara Chakrabarty, Alexander Bridgwater, Axel Lundberg, Filip Skogh, Ahmed Ali-Eldin, David Irwin, and Prashant Shenoy. 2023. CASPER: Carbon-Aware Scheduling and Provisioning for Distributed Web Services. In *Proceedings of the 14th International Green and Sustainable Computing Conference (IGSC), Toronto, ON, Canada*. ACM.

[43] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2020. Energy and Policy Considerations for Modern Deep Learning Research. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. AAAI, New York, NY, 13693–13696.

[44] Thanathorn Sukprasert, Abel Souza, Noman Bashir, David Irwin, and Prashant Shenoy. 2023. Quantifying the Benefits of Carbon-Aware Temporal and Spatial Workload Shifting in the Cloud. arXiv:2306.06502 [cs.DC]

[45] Bo Sun, Russell Lee, Mohammad Hajiesmaili, Adam Wierman, and Danny Tsang. 2021. Pareto-Optimal Learning-Augmented Algorithms for Online Conversion Problems. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 10339–10350.

[46] Bo Sun, Lin Yang, Mohammad Hajiesmaili, Adam Wierman, John CS Lui, Don Towsley, and Danny HK Tsang. 2022. The Online Knapsack Problem with Departures. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 6, 3 (2022), 1–32.

[47] Bo Sun, Ali Zeynali, Tongxin Li, Mohammad Hajiesmaili, Adam Wierman, and Danny H.K. Tsang. 2021. Competitive Algorithms for the Online Multiple Knapsack Problem with Application to Electric Vehicle Charging. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 4, 3, Article 51 (June 2021), 32 pages. https://doi.org/10.1145/3428336

[48] Jennifer Switzer, Gabriel Marcano, Ryan Kastner, and Pat Pannuto. 2023. Junkyard Computing: Repurposing Discarded Smartphones to Minimize Carbon. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. ACM, New York, NY, USA, 400–412.

[49] Swamit Tannu and Prashant J Nair. 2023. The Dirty Secret of SSDs: Embodied Carbon. *ACM SIGENERGY Energy Informatics Review* 3, 3 (2023), 4–9.

[50] John Thiede, Noman Bashir, David Irwin, and Prashant Shenoy. 2023. Carbon Containers: A System-Level Facility for Managing Application-Level Carbon Emissions. In *Proceedings of the 2023 ACM Symposium on Cloud Computing (SoCC '23)*. Association for Computing Machinery, New York, NY, USA, 17–31. https://doi.org/10.1145/3620678.3624644

[51] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. 2015. Large-scale cluster management at Google with Borg. In *Proceedings of the Tenth European Conference on Computer Systems* (Bordeaux, France) *(EuroSys '15)*. Association for Computing Machinery, New York, NY, USA, Article 18, 17 pages. https://doi.org/10.1145/2741948.2741964

[52] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser,

Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272. https://doi.org/10.1038/s41592-019-0686-2

[53] Philipp Wiesner, Ilja Behnke, Dominik Scheinert, Kordian Gontarska, and Lauritz Thamsen. 2021. Let's Wait AWhile: How Temporal Workload Shifting Can Reduce Carbon Emissions in the Cloud. In *Proceedings of the 22nd International Middleware Conference*. Association for Computing Machinery, New York, NY, USA, 260–272. https://doi.org/10.1145/3464298.3493399

[54] Lin Yang, Ali Zeynali, Mohammad H. Hajiesmaili, Ramesh K. Sitaraman, and Don Towsley. 2021. Competitive Algorithms for Online Multidimensional Knapsack Problems. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 5, 3, Article 30 (Dec 2021), 30 pages.

[55] Ali Zeynali, Bo Sun, Mohammad Hajiesmaili, and Adam Wierman. 2021. Data-driven Competitive Algorithms for Online Knapsack and Set Cover. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 12 (May 2021), 10833–10841. https://doi.org/10.1609/aaai.v35i12.17294

[56] ZiJun Zhang, Zongpeng Li, and Chuan Wu. 2017. Optimal posted prices for online cloud resource allocation. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 1, 1 (2017), 1–26.

[57] Jiajia Zheng, Andrew A. Chien, and Sangwon Suh. 2020. Mitigating Curtailment and Carbon Emissions through Load Migration between Data Centers. *Joule* 4, 10 (2020), 2208–2222. https://doi.org/10.1016/j.joule.2020.08.001

[58] Yunhong Zhou, Deeparnab Chakrabarty, and Rajan Lukose. 2008. Budget Constrained Bidding in Keyword Auctions and Online Knapsack Problems. In *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 566–576.

[59] Salah Zrigui, Raphael Y de Camargo, Arnaud Legrand, and Denis Trystram. 2022. Improving the Performance of Batch Schedulers Using Online Job Runtime Classification. *J. Parallel and Distrib. Comput.* 164 (2022), 83–95.

APPENDIX

A DEFERRED PROOFS FROM SECTION 4

A.1 Competitive Proofs for $\texttt{RORO}_{\texttt{cmax}}$

In this section, we provide the proof of Theorem 4.1 that states $\texttt{RORO}_{\text{cmax}}$ for OCSU is α_1-competitive, where α_1 is defined in Equation 9.

Proof of Theorem 4.1. Let $\mathcal{I} \in \Omega$ denote any valid sequence, and let $w^{(j)}$ denote $\texttt{RORO}_{\text{cmax}}$'s final progress before the compulsory execution that begins at time $j \leq T$. Note that $w^{(j)} \in [0, c_{\max}]$. Assuming a job arrives with length c, the worst performance of $\texttt{RORO}_{\text{cmax}}$ happens when $c = c_{\min}$, and $\texttt{RORO}_{\text{cmax}}$ is α_1-competitive. In this scenario, we have two cases:

Case (i). When the job is finished during the compulsory execution.

In this case, $w^{(j)} < c_{\min}$, and according to *lemma B.2* in [25], the offline optimum is lower-bounded by:

$$\texttt{OPT}(\mathcal{I}) \geq c_{\min}(\phi_1(w^{(j)}) - \beta) \tag{10}$$

Additionally, according to *lemma B.3* in [25], the online carbon emissions is upper-bounded by:

$$\texttt{RORO}_{\text{cmax}}(\mathcal{I}) \leq \int_0^{w^{(j)}} \phi_1(u)du + w^{(j)}\beta + (c_{\min} - w^{(j)})U \tag{11}$$

By combining inequalities (10) and (11), we have:

$$\frac{\texttt{RORO}_{\text{cmax}}(\mathcal{I})}{\texttt{OPT}(\mathcal{I})} \leq \frac{\int_0^{w^{(j)}} \phi_1(u)du + w^{(j)}\beta + (c_{\min} - w^{(j)})U}{c_{\min}(\phi_1(w^{(j)}) - \beta)} = \alpha \tag{12}$$

Case (ii). When the job is done without compulsory execution. □

Lemma A.1. *The carbon emissions of* $\texttt{RORO}_{cmax}$ *is upper bounded by:*

$$\texttt{RORO}_{cmax}(\mathcal{I}) \leq c_{min}\phi_1(0) + c_{min}\beta = c_{min}\frac{U}{\alpha} + c_{min}\beta \tag{13}$$

Proof. According to *lemma B.3* in [25], when the job length = c, and the job is done with no compulsory execution, $\texttt{RORO}_{\text{cmax}} \leq \int_0^c \phi_1(u)du + \beta c$. In the worst-case scenario, the job length is $c_{\min}$; hence, we have:

$$\texttt{RORO}_{\text{cmax}}(\mathcal{I}) \leq \int_0^{c_{\min}} \phi_1(u)du + \beta c_{\min} \leq c_{\min}\phi_1(0) + c_{\min}\beta = c_{\min}\frac{U}{\alpha} + 2\beta c_{\min} \tag{14}$$

□

The right hand side of inequality (14) stands correct because $\phi_1(u)$ is monotonically decreasing, and $\phi_1(u) \leq \phi_1(0) = \frac{U}{\alpha} + \beta$.

The lower bound of $\texttt{OPT}(\mathcal{I})$ is:

$$\texttt{OPT}(\mathcal{I}) \geq c_{\min}L \tag{15}$$

Combining inequalities (14) and (15), we have:

$$\frac{\texttt{RORO}_{\text{cmax}}(\mathcal{I})}{\texttt{OPT}(\mathcal{I})} \leq \frac{c_{\min}\frac{U}{\alpha} + 2\beta c_{\min}}{c_{\min}L} = \frac{U}{\alpha L} + \frac{2\beta}{L} = \alpha_{1b} \tag{16}$$

Comparing α_{1b} against α, we note that when $\texttt{RORO}_{\text{cmax}}$ completes the job without compulsory execution, it has *"overspent"* compared to the RORO algorithm which knows the actual job length. This follows by recalling that the threshold functions of both algorithms are monotonically decreasing, and observing their respective threshold values at $w^{(j)} = c_{\min}$, which can be expressed as follows: $\phi_{\text{OCSU}}(c_{\min}) = L+\beta < \phi_1(c_{\min})$. Thus, $\texttt{RORO}_{\text{cmax}}$ allows a strictly higher carbon intensity to run the job with length $c_{\min}$, and the competitive ratio α_{1b} is worse.

Therefore, since $\alpha_1 := \max\{\alpha_{1b}, \alpha\}$, we obtain the following final competitive bound:

$$\alpha_1 = \frac{U}{\alpha L} + \frac{2\beta}{L}$$

A.2 Competitive Proofs for $\texttt{RORO}_{\texttt{cmin}}$

In this section, we provide the proof of Theorem 4.2 that states the instantiation of $\texttt{RORO}_{\text{cmin}}$ for OCSU is α_2-competitive, where $\alpha_2 = \alpha'$ and α' is defined in Equation 8.

Proof of Theorem 4.2. **Roadmap of the proof:** We initially consider an intermediate algorithm, named $\hat{\texttt{ALG}}_2$. This algorithm operates under the assumption that the length of each job is $c_{\min}$ and schedules the job based on a threshold function $\hat{\phi}(\hat{w}), \hat{w} \in [0, c_{\min}]$ (Definition A.2). We first compute this threshold function and the competitive ratio, preparing for the worst-case scenario where the actual job length is $c_{\max}$ (**Step 1**).

Subsequently, we adapt the threshold function $\hat{\phi}(\hat{w})$, scaling it up by a factor of $c_{\max}/c_{\min}$ to obtain a new threshold function $\phi_2(w)$, $w \in [0, c_{\max}]$ (Equation 7), which defines the operation of $\texttt{RORO}_{\text{cmin}}-$ we proceed to determine the competitive ratio (α_2) for $\texttt{RORO}_{\text{cmin}}$, finding that it is less than that of $\hat{\texttt{ALG}}_2$. This outcome reinforces the rationale behind choosing $\texttt{RORO}_{\text{cmin}}$ over $\hat{\texttt{ALG}}_2$, as $\texttt{RORO}_{\text{cmin}}$ proves to be a more strategically sound choice under the given conditions.(**Step 2**)

Step 1: Let $\mathcal{I} \in \Omega$ denote any valid OCSU sequence, and let $\hat{w}^j$ be the $\hat{\texttt{ALG}}_2$'s final progress before the compulsory execution, which begins at time step $j \leq T$, and $\hat{w}^{(j)} \in [0, c_{\min}]$. □

Definition A.2. *Threshold function $\hat{\phi}$ for* OCSU *solved by* $\texttt{RORO}_{cmin}$ *for any progress* $\hat{w} \in [0, c_{min}]$*:*

$$\hat{\phi}(\hat{w}) = U - \beta + \left(\frac{U}{\alpha'} - U + 2\beta\right) \exp\left(\frac{\hat{w}}{c_{min}\alpha'}\right) \tag{17}$$

Lemma A.3. *The offline optimum is lower bounded by* $\texttt{OPT}(\mathcal{I}) \geq (\hat{\phi}(\hat{w}^{(j)}) - \beta)c_{min}$.

Proof. According to *lemma B.2* in [25], the optimal offline strategy, setting aside any additional switching emissions, involves completing the job at the most favorable cost function within the sequence $\{g_t(\cdot)_{t\in[T]}\}$. Suppose the best cost function occurs at an arbitrary step m ($m \in [T], m \leq j$), denoted by $g_m(.)$.

Lemma B.2 in [25] further states that for any job length c, $\texttt{OPT}(\mathcal{I}) = c(\frac{\partial g_m}{\partial x}) \geq c(\hat{\phi}(\hat{w}^{(j)}) - \beta), \quad \hat{w}^{(j)} \in [0, c]$, where $\hat{w}^{(j)}$ is the progress before the compulsory execution. Given that $c \in [c_{\min}, c_{\max}]$ in OCSU problem; the lower bound of OPT is when $c = c_{\min}$, mirroring the assumption made in the $\hat{\texttt{ALG}}_2$ algorithm; therefore, $\texttt{OPT}(\mathcal{I}) \geq (\hat{\phi}(\hat{w}^{(j)}) - \beta)c_{\min}, \quad \hat{w}^{(j)} \in [0, c_{\min}]$ □

Lemma A.4. *The carbon emissions of* $\hat{\texttt{ALG}}_2(\mathcal{I})$ *is upper-bounded by:*

$$\hat{\texttt{ALG}}_2(\mathcal{I}) \leq \int_0^{\hat{w}^{(j)}} \hat{\phi}(u)du + \beta\hat{w}^{(j)} + (c_{max} - \hat{w}^{(j)})U \tag{18}$$

Proof. According to *lemma B.3* in [25], $\texttt{RORO}(\mathcal{I})$ incurred carbon emissions for a job with a length of c is upper-bounded by:

$$\texttt{RORO}(\mathcal{I}) \leq \int_0^{w^{(j)}} \phi_{\texttt{OCS-min}}(u)du + \beta w^{(j)} + (c - w^{(j)})U, w^{(j)} \in [0, c] \tag{19}$$

where $w^{(j)}$ is the progress at time j ($j \leq T$) before the compulsory execution. The term $(c - w^{(j)})U$ denotes the maximum carbon emissions while doing the compulsory execution to satisfy constraint (2). Since $\hat{\texttt{ALG}}_2$ is not aware of the actual job length (c), the maximum carbon emitted during the compulsory execution is $(c_{\max} - \hat{w}^{(j)})U$. Therefore $\hat{\texttt{ALG}}_2(\mathcal{I})$ is upper-bounded by $\int_0^{\hat{w}^{(j)}} \hat{\phi}(u)du + \beta\hat{w}^{(j)} + (c_{\max} - \hat{w}^{(j)})U, :$ $\hat{w}^{(j)} \in [0, c_{\min}]$ □

Combining Lemma A.3 and Lemma A.4 gives:

$$\frac{\hat{\texttt{ALG}}_2(\mathcal{I})}{\texttt{OPT}(\mathcal{I})} \leq \frac{\int_0^{\hat{w}^{(j)}} \hat{\phi}(u)du + \beta\hat{w}^{(j)} + (c_{\max} - \hat{w}^{(j)})U}{(\hat{\phi}(\hat{w}^{(j)}) - \beta)c_{\min}} \leq \alpha' + \frac{U(c_{\max} - c_{\min})}{c_{\min}(\hat{\phi}(\hat{w}^{(j)}) - \beta)} = \alpha_{2a} \tag{20}$$

where the last inequality holds since for any $\hat{w} \in [0, c_{\min}]$:

$$\int_0^{\hat{w}^{(j)}} \hat{\phi}(u)du + \beta\hat{w}^{(j)} + (c_{\max} - \hat{w}^{(j)})U \tag{21}$$

$$= \int_0^{\hat{w}^{(j)}} \left[U - \beta + \left(\frac{U}{\alpha'} - U + 2\beta\right) \exp\left(\frac{\hat{w}^{(j)}}{\alpha' c_{\min}}\right)\right] + \beta\hat{w}^{(j)} + (c_{\max} - \hat{w}^{(j)})U \tag{22}$$

$$= \alpha' c_{\min} \left(\frac{U}{\alpha'} - U + 2\beta\right) \left[\exp\left(\frac{w}{\alpha' c_{\min}}\right) - 1\right] + \beta\hat{w}^{(j)} + (c_{\max} - \hat{w}^{(j)})U + (U - \beta)\hat{w}^{(j)} \tag{23}$$

$$= \alpha' c_{\min} \left(\frac{U}{\alpha'} - U + 2\beta\right) \left[\exp\left(\frac{w}{\alpha' c_{\min}}\right) - 1\right] + Uc_{\max} \tag{24}$$

$$= \alpha' c_{\min} \left[U - 2\beta + \left(\frac{U}{\alpha'} + 2\beta - U\right) \exp\left(\frac{\hat{w}^{(j)}}{\alpha' c_{\min}}\right)\right] + U(c_{\max} - c_{\min}) \tag{25}$$

$$= \alpha' c_{\min}(\hat{\phi}(\hat{w}^{(j)}) - \beta) + U(c_{\max} - c_{\min}) \tag{26}$$

In what follows, we will calculate the optimal α' (as in [25, Theorem 3.3]) to define the threshold function based on the assumptions established for $\hat{\texttt{ALG}}_2$ (namely, that $\hat{\texttt{ALG}}_2$ assumes the job has length $c_{\min}$ and that it must complete the job during the compulsory execution if it has length $> c_{\min}$). It is known that worst-case instances for online search problems such as OCSU occur when inputs arrive in decreasing order of cost (i.e., carbon intensity) [25, 45, 47]. We formalize these *x-instances* below.

Definition A.5 (x-instance for OCSU). *For sufficiently large $m, n \in \mathbb{Z}$, we let $\delta := (U-L)/m$. Given $x \in [L, U]$, $\mathcal{I}_x \in \Omega$ is an x-instance for* OCSU *which consists of $m_x := 2\lceil (x-L)/\delta \rceil + 1$ alternating blocks of cost functions. For $i \in [m_x - 2]$, the ith block contains n linear cost functions with coefficient U if i is odd, or a single linear cost function with coefficient $U - \lceil i/2 \rceil \delta$ when i is even. The last 2 blocks consist of n linear cost function with coefficients $(x + \epsilon)$, followed by n cost functions with coefficients U.*

As $m \to \infty$, the alternating blocks of single "good" cost functions continuously decrease down to x, and each of these blocks is interrupted by a long block of worst-case U functions. Note that $\mathcal{I}_U$ is a simple stream of n cost functions, all with coefficient U, and that the last cost function for any $\mathcal{I}_x$ are always U (i.e., the marginal emission is maximized during the compulsory execution).

Lemma A.6. *Any deterministic online algorithm* ALG *for* OCSU *which assumes the job has length c_{min} (and is thus forced to complete longer jobs during the compulsory execution) has a competitive ratio of at least α' (where α' is as defined in* (33) *).*

Proof. On any x-instance $\mathcal{I}_x$, we may fully describe the actions of any deterministic algorithm ALG via a conversion function $h(x) : [L, U] \to [0, c_{\min}]$. Note that this function is unidirectional (irrevocable), and non-increasing in $[L, U]$ such that $h(x - \delta) \geq h(x)$, since processing $\mathcal{I}_{x-\delta}$ is equivalent to first processing $\mathcal{I}_x$ (besides the final two blocks) and then processing blocks with marginal emissions of $x - \delta$ and U. The total emission of ALG described by the conversion function $h(x)$ on instance $\mathcal{I}_x$ is expressed as follows:

$$\texttt{ALG}(\mathcal{I}_x) = h(U/\alpha')U/\alpha' - \int_{U/\alpha'}^{x} udh(u) + (c - h(x))U \leq h(U/\alpha')U/\alpha' - \int_{U/\alpha'}^{x} udh(u) + (c_{\max} - h(x))U \tag{27}$$

We note that on an instance $\mathcal{I}_x$, $\texttt{OPT}(I_x) \to c_{\max} x$ as $\epsilon \to 0$ and n is sufficiently large. Letting ALG be α'-competitive, we then have the following necessary condition on the conversion function when considering equation (27):

$$h(U/\alpha')U/\alpha' - \int_{U/\alpha'}^{x} udh(u) + (c_{\max} - h(x))U \leq \alpha' c_{\max} x \tag{28}$$

By integral by parts, (28) implies that $h(x)$ must satisfy:

$$h(x) \geq \frac{c_{\max} U - \alpha' c_{\max} x}{U - x - 2\beta} + \frac{1}{U - x - 2\beta} \int_{U/\alpha'}^{x} h(u)du \tag{29}$$

By Gronwall's Inequality [34, p. 356, Theorem 1], we have:

$$h(x) \geq \frac{U c_{\max} - \alpha' c_{\max} x}{U - x - 2\beta} + \left[\frac{U\alpha' c_{\max} - U c_{\max} - 2\beta c_{\max}}{u + 2\beta - U} - \alpha' c_{\max} \ln(u + 2\beta - U) \right]_{U/\alpha'}^{x} \tag{30}$$

$$h(x) \geq \alpha' c_{\max} \ln(U/\alpha' + 2\beta - U) - \alpha' c_{\max} \ln(x + 2\beta - U) \tag{31}$$

By the problem definition, the job with length $c_{\min}$ should be completed upon observing the best carbon intensity L, i.e., $h(x) \leq h(L) \leq c_{\min}$, giving the following:

$$c_{\min}/c_{\max} \geq \alpha' \ln(U/\alpha' + 2\beta - U) - \alpha' \ln(L + 2\beta - U). \tag{32}$$

The optimal α' is obtained when the above inequality is binding, which gives the following:

$$\alpha' = \left[\frac{c_{\max}}{c_{\min}} W \left[\frac{c_{\min}}{c_{\max}} \left(\frac{2\beta}{U} + \frac{L}{U} - 1 \right) \exp\left(\frac{c_{\min}}{c_{\max}} \left(\frac{2\beta}{U} - 1 \right) \right) \right] - \frac{2\beta}{U} + 1 \right]^{-1}. \tag{33}$$

□

Step 2: By scaling up $\phi(\hat{w}), w \in [0, c_{\min}]$ to the factor of $c_{\max}/c_{\min}$, we will have $\phi_2(w), w \in [0, c_{\max}]$ in (7). The competitive ratio of $\texttt{RORO}_{\text{cmin}}$ that utilizes $\phi_2(w)$ can be derived using the following two cases:

• **If $\texttt{RORO}_{\text{cmin}}$ completes any amount of the job before the compulsory execution.** In this case, the analysis from Section 4 exactly translates to the $\texttt{RORO}_{\text{cmin}}$ setting. Substituting ϕ_2 for ϕ_1 gives the following competitive bound for this case:

$$\frac{\texttt{RORO}_{\text{cmin}}(\mathcal{I})}{\texttt{OPT}(\mathcal{I})} = \frac{U}{\alpha' L} + \frac{2\beta}{L}. \tag{34}$$

• **If $\texttt{RORO}_{\text{cmin}}$ completes none of the job before the compulsory execution.** In this case, we know that $\texttt{OPT}(\mathcal{I})$ is lower-bounded by $\phi_2(0) - \beta$, because if a cost function better than $\phi_2(0) - \beta$ arrived during the instance $\mathcal{I}$, $\texttt{RORO}_{\text{cmin}}$ would have completed a non-zero amount of the job before the compulsory execution. This gives the following competitive bound:

$$\frac{\texttt{RORO}_{\text{cmin}}(\mathcal{I})}{\texttt{OPT}(\mathcal{I})} = \frac{U c_{\min}}{[\phi_2(0) - \beta]\, c_{\min}} = \frac{U}{U/\alpha'} = \alpha'. \tag{35}$$

Because α' approaches U/L as $c_{\max}/c_{\min}$ grows, the competitive ratio in the second case is the worst-case bound, yielding the following competitive bound for $\texttt{RORO}_{\text{cmin}}$:

$$\alpha_2 = \alpha'. \tag{36}$$

Here we note that α_2 does not contain an extra linear dependence on $c_{\max}/c_{\min}$ which *is* present in α_{2a}, implying that $\alpha_2 \leq \alpha_{2a}$. This is intuitive, since even if $\hat{\texttt{ALG}}_2$ completes some fraction of the job before the compulsory execution, it must complete $(c_{\max} - c_{\min})$ of the job during the compulsory execution, whereas the scaled threshold in $\texttt{RORO}_{\text{cmin}}$ allows it to be more flexible. In the rest of the paper, we use the design of $\texttt{RORO}_{\text{cmin}}$ as our baseline based on its improved theoretical bounds and its superior performance in experiments.

A.3 Analyzing the impact of rate constraints

We note that the rate constraint $d_t : t \in [T]$ surprisingly does not appear in the worst-case analysis of $\texttt{RORO}_{\text{cmax}}$ and $\texttt{RORO}_{\text{cmin}}$. In this section, we give intuitive justification to explain this dynamic for completeness.

Specifically, we show that when a threshold-based algorithm (i.e., $\texttt{RORO}_{\text{cmax}}$ or $\texttt{RORO}_{\text{cmin}}$) achieves a certain competitive ratio for OCSU when $d_t = c \ \forall t \in [j]$ (i.e., when the rate allows completing the entire job in a single time slot), the worst-case competitive ratio will stay constant if $d_t < c \ \forall t \in [j]$.

Lemma A.7. *Let* ALG *denote a threshold-based algorithm for* OCSU *which uses a threshold function* $\psi(w)$. *Suppose* ALG *is* η*-competitive when* $d_t = c \ \forall t \in [j]$. *If* $d_t < c \ \forall t \in [j]$, *the competitive ratio of* ALG *is still upper bounded by* η.

Proof. We consider the case where a rate constraint $d_t < c$ causes ALG to make a decision which violates its worst-case competitive ratio of η. At any arbitrary time slot t, the disconnect between the setting where $x_t \in [0, c]$ and the setting with rate constraints $< c$, where $x_t \in [0, d_t]$, is that x_t cannot be $> d_t$. Intuitively, a challenging situation for ALG under such a constraint is the case where ALG would otherwise run more than $> d_t$ of the job during a period of "good" carbon intensity (before the compulsory execution), but it is now constrained from doing so.

We now show that such a situation implies that ALG achieves a worst-case competitive ratio which is equal to or better than η. Recall that $w^{(t)}$ denotes the progress of ALG after time step t.

For an instance $\mathcal{I} \in \Omega$ and an arbitrary time step m, let $w^{(m)} = w^{(m-1)} + d_m$, implying that $x_m = d_m$. For the sake of comparison, we first consider this time step with a cost function $g_m(\cdot)$ such that $g_m(x_m) + \beta|x_m - x_{m-1}| = \psi(w^{(m)})x_m$, implying that even if the rate constraint was not present, ALG would set $x_m = d_m$. If no more of the job is completed by ALG after time step m (ignoring the compulsory execution), we know that ALG is η-competitive (e.g., for $\texttt{RORO}_{\text{cmax}}$, $\eta = \alpha_1$, and for $\texttt{RORO}_{\text{cmin}}$, $\eta = \alpha_2$).

Consider the exact same setting, except with a substituted cost function $g'_m(\cdot)$, such that $g'_m(x_m) + \beta|x_m - x_{m-1}| < \psi(w^{(m)}) \cdot x_m$. We denote this new instance by $\mathcal{I}'$. This implies that without the presence of a rate constraint, ALG would set $x_m > d_m$. In other words, $g'_m(\cdot)$ has a "good carbon intensity", but ALG cannot run as much of the job as it otherwise should due to the rate constraint.

Note that OPT is also subject to the same rate constraint d_m. Thus, we know that $\texttt{OPT}(\mathcal{I}')$ is lower bounded by $[\psi(w^{(m)}) - \beta](1 - d_m) + g'_m(d_m)$ – the rest of the optimal solution is bounded by the final threshold value, since we assume that no more of the job is completed by ALG after time step m.

The worst-case carbon emission of ALG is upper bounded by $\texttt{ALG}(\mathcal{I}') \leq \texttt{ALG}(\mathcal{I}) - \int_{w^{(m-1)}}^{w^{(m)}} \psi(u)du + g'_m(d_m)$, which follows since we substitute the last portion of the threshold function (of "width" d_m) with the new cost function $g'_m(d_m)$.

Compared to the previous setting of $\mathcal{I}$, the OPT and ALG solutions have both decreased – $\texttt{OPT}(\mathcal{I}')$ has decreased by a factor of $g'_m(d_m) - [\psi(w^{(m)}) - \beta]d_m$, while $\texttt{ALG}(\mathcal{I}')$ has decreased by a factor of $g'_m(d_m) - \int_{w^{(m-1)}}^{w^{(m)}} \psi(u)du$.

However, note that since ψ is monotonically decreasing in w, by definition, $[\psi(w^{(m)}) - \beta]d_m < \int_{w^{(m-1)}}^{w^{(m)}} \psi(u)du$. Thus, the cost of ALG has improved *more* than the cost of OPT. This then implies the following:

$$\frac{\texttt{ALG}(\mathcal{I})}{\texttt{OPT}(\mathcal{I})} \leq \frac{\texttt{ALG}(\mathcal{I}) - \int_{w^{(m-1)}}^{w^{(m)}} \psi(u)du + g'_m(d_m)}{[\psi(w^{(m)}) - \beta](c - d_m) + g'_t(d_m)} < \eta.$$

At a high-level, this result shows that even if there is a rate constraint which prevents ALG from accepting a good carbon intensity, the worst-case competitive ratio does not change. □

A.4 LACS Consistency and Robustness for OCSU

In the following, we prove Theorem 4.3, which states that the instantiation of LACS for OCSU is $(\alpha + \gamma)$-consistent and $\left[\left(1 - \frac{\gamma}{\alpha_1 - \texttt{sign}(\alpha_1 - \alpha_2)\epsilon - \alpha}\right)\alpha^{\max}_{\texttt{RORO}_{\text{pred}}} + \left(\frac{\gamma(\alpha_1 - \texttt{sign}(\alpha_1 - \alpha_2)\epsilon)}{\alpha_1 - \texttt{sign}(\alpha_1 - \alpha_2)\epsilon - \alpha}\right)\right]$-robust. We note that consistency and robustness in learning-augmented algorithm design describe an algorithm's performance when the predictions are exactly accurate and entirely incorrect, respectively. See Definition 2.2 for the formal definitions of both consistency and robustness.

Proof of Theorem 4.3. Initially, we start by noting that the online solutions given by $\texttt{RORO}_{\text{robust}}$ and LACS are always feasible considering the constraint in Equation 2. Let $\mathcal{I} \in \Omega$ be an arbitrary valid OCSU sequence, and for a job with length c:

$$\texttt{RORO}_{\text{robust}}(\mathcal{I}) : \sum_{t=1}^{T} \tilde{x}_t = \sum_{t=1}^{T} [kx_{1t} + (1-k)x_{2t}] > kc + (1-k)c > c$$

$$\texttt{LACS}(\mathcal{I}) : \sum_{t=1}^{T} x_t = \sum_{t=1}^{T} [\lambda \hat{x}_t + (1-\lambda)\tilde{x}_t] > \lambda c + (1-\lambda)c > c$$

□

Lemma A.8. *The carbon emissions of* $\texttt{RORO}_{robust}$ *is bounded by:*

$$\texttt{RORO}_{robust}(\mathcal{I}) \leq k\texttt{RORO}_{cmax}(\mathcal{I}) + (1-k)\texttt{RORO}_{cmin}(\mathcal{I}) \tag{37}$$

Proof.

$$\begin{aligned}
\texttt{RORO}_{\text{robust}}(\mathcal{I}) &= \sum_{t=1}^{T} g_t(\tilde{x}_t) + \sum_{t=1}^{T+1} \beta|\tilde{x}_t - \tilde{x}_{t-1}| \\
&= \sum_{t=1}^{T} g_t(kx_{1t} + (1-k)x_{2t}) + \sum_{t=1}^{T+1} \beta|kx_{1t} + (1-k)x_{2t} - kx_{2(t-1)} - (1-k)x_{2(t-1)}| \\
&\leq k\sum_{t=1}^{T} g_t(x_{1t}) + (1-k)\sum_{t=1}^{T} g_t(x_{2t}) + \sum_{t=1}^{T+1} \beta|kx_{1t} - kx_{1(t-1)}| + \sum_{t=1}^{T+1} \beta|(1-k)x_{2t} - (1-k)x_{2(t-1)}| \\
&\leq k(\sum_{t=1}^{T} g_t(x_{1t}) + \beta|x_{1t} - x_{1(t-1)}|) + (1-k)(\sum_{t=1}^{T} g_t(x_{2t}) + \beta|x_{2t} - x_{2(t-1)}|) \\
&\leq k\texttt{RORO}_{\text{cmax}}(I) + (1-k)\texttt{RORO}_{\text{cmin}}(I)
\end{aligned}$$

□

Since $\texttt{RORO}_{\text{cmax}}(\mathcal{I}) \leq \alpha_1 \texttt{OPT}(\mathcal{I})$ and $\texttt{RORO}_{\text{cmin}}(\mathcal{I}) \leq \alpha_2 \texttt{OPT}(\mathcal{I})$ by definition, we have:

$$\texttt{RORO}_{\text{robust}}(\mathcal{I}) \leq (k\alpha_1 + (1-k)\alpha_2)\texttt{OPT}(\mathcal{I}) \tag{38}$$

$$\texttt{RORO}_{\text{robust}}(\mathcal{I}) \leq (k(\alpha_1 - \alpha_2) + \alpha_2)\texttt{OPT}(\mathcal{I}) \tag{39}$$

We denote $\epsilon \in [0, |\alpha_1 - \alpha_2|]$, and we set $k = 1 - \frac{\epsilon}{|\alpha_1 - \alpha_2|}$; therefore, we have:

$$\texttt{RORO}_{\text{robust}}(\mathcal{I}) \leq (\alpha_1 - \texttt{sign}(\alpha_1 - \alpha_2)\epsilon)\texttt{OPT}(\mathcal{I}) = \alpha_{\texttt{RORO}_{\text{robust}}}\texttt{OPT}(\mathcal{I}) \tag{40}$$

where $\texttt{sign}(x)$ is the sign function.

By using the same proof in Lemma A.8, we can show that:

$$\texttt{LACS}(\mathcal{I}) \leq \lambda\texttt{RORO}_{\text{pred}}(\mathcal{I}) + (1-\lambda)\texttt{RORO}_{\text{robust}}(\mathcal{I}) \tag{41}$$

Lemma A.9. LACS *is* $(\alpha + \gamma)$*-consistent with accurate predictions.*

Proof. We assume $\texttt{RORO}_{\text{pred}}(\mathcal{I})$ has the perfect prediction of the job length ($\hat{c} = c$), and by leveraging the perfect prediction (excepting minor differences in the compulsory execution), we have that $\texttt{RORO}_{\text{pred}}(\mathcal{I}) = \texttt{RORO}(\mathcal{I})$. Therefore, by definition, $\texttt{RORO}_{\text{pred}}(\mathcal{I}) \leq \alpha\texttt{OPT}(\mathcal{I})$

Considering Equation 40 and Equation 41, we have:

$$\texttt{LACS}(\mathcal{I}) \leq \lambda\texttt{RORO}_{\text{pred}}(\mathcal{I}) + (1-\lambda)\texttt{RORO}_{\text{robust}}(\mathcal{I}) \tag{42}$$

$$\texttt{LACS}(\mathcal{I}) \leq \lambda\alpha\texttt{OPT}(\mathcal{I}) + (1-\lambda)\alpha_{\texttt{RORO}_{\text{robust}}}\texttt{OPT}(\mathcal{I}) \tag{43}$$

$$\texttt{LACS}(\mathcal{I}) \leq (\lambda\alpha + (1-\lambda)\alpha_{\texttt{RORO}_{\text{robust}}})\texttt{OPT}(\mathcal{I}) \tag{44}$$

$$\texttt{LACS}(\mathcal{I}) \leq (\lambda\alpha + (1-\lambda)(\alpha_1 - \texttt{sign}(\alpha_1 - \alpha_2)\epsilon))\texttt{OPT}(\mathcal{I}) \tag{45}$$

Since $\alpha \leq \alpha_1 - \texttt{sign}(\alpha_1 - \alpha_2)\epsilon \quad \forall \epsilon \in [0, |\alpha_1 - \alpha_2|]$, and we have $\gamma \in [0, \alpha_1 - \texttt{sign}(\alpha_1 - \alpha_2)\epsilon - \alpha]$; therefore we set $\lambda = 1 - \frac{\gamma}{\alpha_1 - \texttt{sign}(\alpha_1 - \alpha_2)\epsilon - \alpha}$, and we have:

$$\texttt{LACS}(\mathcal{I}) \leq (\alpha + \gamma)\texttt{OPT}(\mathcal{I}) \tag{46}$$

□

Lemma A.10. LACS *is* $\left(\left(1 - \frac{\gamma}{\alpha_1 - \texttt{sign}(\alpha_1 - \alpha_2)\epsilon - \alpha}\right)\alpha^{\max}_{\texttt{RORO}_{pred}} + \left(\frac{\gamma(\alpha_1 - \texttt{sign}(\alpha_1 - \alpha_2)\epsilon)}{\alpha_1 - \texttt{sign}(\alpha_1 - \alpha_2)\epsilon - \alpha}\right)\right)$*-robust for any prediction.*

Proof. To calculate the competitive ratio of $\texttt{RORO}_{\text{pred}}(\mathcal{I})$ when the job length prediction error is maximized, we consider two cases: **Case (i)** $[\hat{c} = c_{\min}, c = c_{\max}]$: Since $\texttt{RORO}_{\text{pred}}$ assumes the job length is $c_{\min}$ it will utilize the threshold function below:

$$\phi(w) = U - \beta + \left(\frac{U}{\alpha} - U + 2\beta\right) \exp\left(\frac{w}{c_{\min}\alpha}\right) \tag{47}$$

Let $w^{(j)}$ be the final progress before the compulsory execution at time $j \leq T$, and let $\mathcal{I} \in \Omega$ be a OCSU sequence that the minimum carbon intensity L is revealed at time $m \leq j \leq T$. By disregarding extra switching emissions, $\texttt{OPT}(\mathcal{I}) \longrightarrow c_{\max}L$. In [25], by the definition of the threshold function for any job length, we use the threshold function for $c_{\min}$ in Equation 47, when the minimum carbon intensity L arrives at time m, the remained amount of the job until $w^{(j)}$ would be scheduled; hence, $w^m = w^{(j)}$ and $m = j$, and according to *Lemma B.2* in [25], $L = \phi(w^{(j)}) - \beta$ which means no other carbon intensities are accepted and the rest of the job $(c_{\min} - w^{(j)})$ should be done during compulsory execution. By *Lemma B.3* in [25] and observing that the rest of the job must be completed in the compulsory execution, $\texttt{RORO}_{\text{pred}}(\mathcal{I})$ is upper-bounded by:

$$\texttt{RORO}_{\text{pred}}(\mathcal{I}) \leq \left[\int_0^{w^{(j)}} \phi(w) + \beta w^{(j)} + (c_{\min} - w^{(j)})U\right] + (c_{\max} - c_{\min})U \tag{48}$$

$$\leq [\alpha(\phi(w^{(j)}) - \beta] + (c_{\max} - c_{\min})U \tag{49}$$

$$\leq \alpha L + (c_{\max} - c_{\min})U \tag{50}$$

The term $(c_{\max} - c_{\min})U$ in (48) is the amount of remaining job that must be done during compulsory execution since $c = c_{\max}$.
Considering (50) and the lower bound of $\texttt{OPT}(\mathcal{I})$, we have:

$$\frac{\texttt{RORO}_{\text{pred}}(\mathcal{I})}{\texttt{OPT}(\mathcal{I})} \leq \frac{\alpha L + (c_{\max} - c_{\min})U}{c_{\max}L} \leq \frac{\alpha}{c_{\max}} + \frac{c_{\max} - c_{\min}}{c_{\max}}\frac{U}{L} = \alpha^1_{\texttt{RORO}_{\text{pred}}} \tag{51}$$

Case (ii) $[\hat{c} = c_{\max}, c = c_{\min}]$: In this case, $\texttt{RORO}_{\text{pred}}$ uses the exact same threshold function as $\texttt{RORO}_{\text{cmax}}$. Thus, $\texttt{RORO}_{\text{cmax}}(\mathcal{I}) = \texttt{RORO}_{\text{pred}}(\mathcal{I})$, and we inherit the following competitive bound:

$$\frac{\texttt{RORO}_{\text{pred}}(\mathcal{I})}{\texttt{OPT}(\mathcal{I})} \leq \frac{U}{\alpha L} + \frac{2\beta}{L} = \alpha^2_{\texttt{RORO}_{\text{pred}}} \tag{52}$$

Considering both **Case (i)** and **Case (ii)**, we let $\alpha^{\max}_{\texttt{RORO}_{\text{pred}}} = \max\{\alpha^1_{\texttt{RORO}_{\text{pred}}}, \alpha^2_{\texttt{RORO}_{\text{pred}}}\}$ to reflect the worst-case in either of these cases. By Equation 41, we have the following robustness bound:

$$\texttt{LACS}(\mathcal{I}) \leq \lambda\texttt{RORO}_{\text{pred}}(\mathcal{I}) + (1-\lambda)\texttt{RORO}_{\text{robust}}(\mathcal{I}) \tag{53}$$

$$\texttt{LACS}(\mathcal{I}) \leq \lambda\alpha^{\max}_{\texttt{RORO}_{\text{pred}}}\texttt{OPT}(\mathcal{I}) + (1-\lambda)\alpha_{\texttt{RORO}_{\text{robust}}}\texttt{OPT}(\mathcal{I}) \tag{54}$$

$$\texttt{LACS}(\mathcal{I}) \leq (\lambda\alpha^{\max}_{\texttt{RORO}_{\text{pred}}} + (1-\lambda)(\alpha_1 - \epsilon))\texttt{OPT}(\mathcal{I}) \tag{55}$$

$$\texttt{LACS}(\mathcal{I}) \leq \left(\left(1 - \frac{\gamma}{\alpha_1 - \texttt{sign}(\alpha_1 - \alpha_2)\epsilon - \alpha}\right)\alpha^{\max}_{\texttt{RORO}_{\text{pred}}} + \left(\frac{\gamma(\alpha_1 - \texttt{sign}(\alpha_1 - \alpha_2)\epsilon)}{\alpha_1 - \texttt{sign}(\alpha_1 - \alpha_2)\epsilon - \alpha}\right)\right)\texttt{OPT}(\mathcal{I}) \tag{56}$$

□

By combining the results of Lemma A.9 and Lemma A.10, the statement of Theorem 4.3 follows, and we conclude that LACS for OCSU is $(\alpha + \gamma)$-consistent and $\left[\left(1 - \frac{\gamma}{\alpha_1 - \texttt{sign}(\alpha_1-\alpha_2)\epsilon - \alpha}\right)\alpha^{\max}_{\texttt{RORO}_{\text{pred}}} + \left(\frac{\gamma(\alpha_1 - \texttt{sign}(\alpha_1-\alpha_2)\epsilon)}{\alpha_1 - \texttt{sign}(\alpha_1-\alpha_2)\epsilon - \alpha}\right)\right]$-robust.