Data-driven Algorithm Selection for Carbon-Aware Scheduling

Roozbeh Bostandoost
University of Massachusetts Amherst
USA

Walid A. Hanafy
University of Massachusetts Amherst
USA

Adam Lechowicz
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

As computing demand continues to grow, minimizing its environmental impact has become crucial. This paper presents a study on carbon-aware scheduling algorithms, focusing on reducing carbon emissions of delay-tolerant batch workloads. Inspired by the Follow the Leader strategy, we introduce a simple yet efficient meta-algorithm, called FTL, that dynamically selects the most efficient scheduling algorithm based on real-time data and historical performance. Without fine-tuning and parameter optimization, FTL adapts to variability in job lengths, carbon intensity forecasts, and regional energy characteristics, consistently outperforming traditional carbon-aware scheduling algorithms. Through extensive experiments using real-world data traces, FTL achieves 8.2% and 14% improvement in average carbon footprint reduction over the closest runner-up algorithm and the carbon-agnostic algorithm, respectively, demonstrating its efficacy in minimizing carbon emissions across multiple geographical regions.

CCS CONCEPTS

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

KEYWORDS

Sustainable computing, data-driven algorithm, carbon-aware scheduling, temporal shifting

ACM Reference Format:
Roozbeh Bostandoost, Walid A. Hanafy, Adam Lechowicz, Noman Bashir, Prashant Shenoy, and Mohammad Hajiesmaili. 2024. Data-driven Algorithm Selection for Carbon-Aware Scheduling. In *Proceedings of 3rd Workshop on Sustainable Computer Systems (HotCarbon'24).* ACM, New York, NY, USA, 6 pages.

1 INTRODUCTION

The demand for computing has increased exponentially, propelling energy consumption to unprecedented levels [1]. This surge in energy use not only exacerbates the carbon footprint associated with electricity generation but also emphasizes the urgent need for sustainable computing practices. For instance, computing's carbon footprint has increased by 5% from 2015 to 2020 [13]. Among various strategies proposed to mitigate these effects, carbon-aware computing has emerged as a critical research area, where the focus is on modulating the power consumption of data centers in alignment with renewable energy availability and carbon intensity of the electricity grid. Techniques such as temporal shifting, spatial shifting, resource scaling, and dynamic voltage and frequency scaling (DVFS) have been explored to minimize carbon emissions [3, 6–8, 10, 11, 18–20]. This paper focuses on carbon-aware temporal shifting of delay-tolerant batch workloads (e.g., ML training jobs and MPI simulations), leveraging the variability in energy's carbon intensity by shifting computations from high- to low-carbon periods.

The effectiveness of temporal shifting strategies hinges on the ability of scheduling algorithms to adapt to temporal fluctuations in carbon intensity. Existing research in this domain has introduced a variety of algorithms designed to optimize scheduling in response to these dynamics [10, 11, 15, 18, 20]. However, these algorithms can perform inconsistently across different operational contexts due to varying local energy market conditions and the unpredictable nature of renewable energy sources. This inconsistency presents a significant challenge: *no single configuration of an algorithm, or even a single algorithm reliably outperforms others in all situations.* Each algorithm's performance can drastically change with variations in carbon intensity and job characteristics, making static algorithm selection strategies suboptimal. We refer to Section 3.2 for detailed measurement of the performance of different algorithms across different settings.

Observing that no algorithm is superior to others, in this paper, and motivated by the emerging topic of data-driven algorithm selection [2, 5, 21], we propose a meta-algorithm, FTL inspired by the "**F**ollow **t**he **L**eader" strategy [9] that addresses the variability in performance of temporal shifting algorithms through an empirical, data-driven approach. Unlike traditional methods that rely on static or heuristic-based decision-making, FTL dynamically selects the most efficient algorithm based on real-time data and historical performance metrics. It considers various factors, including the predicted length of jobs, the historical effectiveness of scheduling algorithms within similar operational windows, and carbon intensity forecasts for specific geographical regions. This approach allows the scheduler to adaptively choose the most suitable algorithm for any given situation, enhancing the carbon efficiency of computing operations.

To develop FTL, we conduct a comprehensive empirical analysis spanning diverse datasets from multiple geographical regions over three years (2020-2022). By systematically analyzing the performance of various scheduling algorithms across different scenarios, we gather insights into their operational efficacy relative to fluctuating carbon intensity levels. This analysis informs the design of FTL, which uses a decision-making framework to select the optimal scheduling algorithm based on a comparative assessment of the historical performance of scheduling algorithms.

Contributions: The main contributions of this paper are as follows.

(1) We conduct a detailed analysis of the carbon reduction of a broad range of carbon-aware scheduling algorithms, considering various job characteristics and regions. Our findings underscore the necessity for an adaptive meta-algorithm that can respond dynamically to changing conditions.
(2) We introduce a meta-algorithm, FTL, inspired by the "Follow the Leader" strategy. This algorithm intelligently selects the optimal scheduling algorithm for each job based on historical performance data of scheduling algorithms and predicted job length.
(3) Through comprehensive experiments, we show that FTL consistently surpasses individual carbon-aware scheduling algorithms in reducing carbon emissions across diverse regions and settings. Specifically, FTL achieves 8.2% and 14% average carbon reductions over the closest runner-up algorithm and carbon-agnostic scheduling, respectively.

2 PROBLEM STATEMENT

We introduce the temporal shifting problem as the Online Carbon-Aware Scheduling (OCS) problem, which entails completing a job of unknown but bounded total length $c \in [c_{\mathtt{min}}, c_{\mathtt{max}}]$ (where $c_{\mathtt{min}}$ and $c_{\mathtt{max}}$ are known) while minimizing the carbon emissions of execution. In this setting, a job arrives at the scheduler without a known length, and the scheduler must make decisions over a set time frame, from $t = 1$ to T, where T represents the service-level objective (deadline) for the job's completion.

At each time slot t, the scheduler receives the current carbon intensity, denoted as C_t (e.g., in gCO$_2$eq./kWh), and decides whether to execute one unit of the job—or any remaining fraction thereof—or to pause the execution. This decision is represented by the binary variable x_t, where $x_t = 1$ indicates resuming the job and $x_t = 0$ means pausing it. The carbon emissions associated with operating at time t are calculated as $C_t \times E \times x_t$[1], where E represents the energy consumption per unit of time (e.g., in kWh). The job completion constraint enforces that the job must be completed by the deadline, i.e., $\sum_{t\in[T]} x_t \geq c$. Without knowledge of the remaining job length, an algorithm must assume that at any time i, there are $c_{\mathtt{max}} - w_i$ units left to schedule, where w_i represents the completed job units up to i. This motivates a *compulsory execution* strategy when only $T - (c_{\mathtt{max}} - w_i)$ time slots remain to avoid violating the job completion constraint.

Furthermore, any change in the allocation decision (pause or resume) between consecutive time steps incurs additional carbon emissions due to the energy overhead of checkpointing or restoring to save or retrieve the job's state, respectively. The overhead of such operations depends on the frequency of checkpoint and restore actions and the energy cost of a single operation. These operations are influenced by the size of the job's state as discussed in [17]. The extra emissions from a single checkpoint or restore are quantified as $\beta \times E \times C_t$, where β is a linear coefficient indicating the fraction of a time unit consumed by these operations. We assume that β is known to the scheduler as it is highly correlated with the job memory requirements [16].

The offline objective of OCS, summarized below, involves minimizing both execution and switching carbon emissions:

$$\mathtt{OCS}: \min_{\{x_t\}_{t\in[T]}} \underbrace{\sum_{t=1}^{T} C_t \times E \times x_t}_{\text{Execution carbon emissions}} + \underbrace{\sum_{t=1}^{T+1} \beta E C_t |x_t - x_{t-1}|}_{\text{Switching carbon emissions}} \quad (1)$$

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

In this paper, we focus on the dynamic setting of OCS, where the actual job length c is revealed only upon satisfying the job completion constraint in Equation 2 (i.e., once the job is finished).

3 DESIGNING FTL META-ALGORITHM

In this section, we first review existing algorithms that tackle the temporal shifting problem. We then explore the variations in performance among these algorithms, highlighting the lack of a definitive "best algorithm" for all scenarios. Motivated by this finding, we present FTL, a meta-algorithm inspired by the **F**ollow **t**he **L**eader strategy [9]. This approach intelligently selects the most suitable algorithm for each incoming job based on its predicted length, $\hat{c}$, ensuring optimal performance tailored to the specific conditions of each job and region.

3.1 Baseline Algorithms

Single Threshold Algorithm: The Single Threshold algorithm, inspired by theoretical literature on online search [4], employs a fixed threshold to manage job execution based on carbon intensity. At each time step t, it checks if the carbon intensity is below a certain threshold. If the intensity is low enough, the job will run; if not, it will pause. This approach does not assume any knowledge of job length information and does not consider the overhead of switching. We denote variations of this algorithm as ST$[p]$, where p represents different threshold settings.

Double-Threshold Algorithm: Motivated by the Single Threshold algorithm's neglect of switching overhead, Lechowicz et al. [11] present a Double-Threshold algorithm for temporal shifting. This algorithm employs two distinct thresholds—a low and a high threshold, separated by a margin, α. Its operation involves two key conditions: if the job was paused at the previous time step, the carbon intensity must decrease to at least α below the high threshold, ensuring it is less than or equal to the low threshold before resuming operation. This strategy ensures that the emissions incurred from restoring the job are outweighed by the benefits of operating at a lower carbon intensity. Conversely, if the job was active at the previous time step, it will continue to run unless the carbon intensity climbs to at least α units above the low threshold (i.e., the high threshold). This approach helps to avoid frequent checkpointing/restoring and the associated emissions from repeatedly stopping and restarting the job, minimizing

[1]If a job is completed before utilizing the entire time unit, we proportionally account for the fractional carbon consumption incurred during that period.

Table 1: Number of times an algorithm achieves the lowest carbon consumption for medium switching overhead tasks by region (expressed as %)

Algorithm \ Region	Ontario	California	AU-NSW	France
`DT[40, 60]`	18.46%	22.38%	24.15%	14.77%
`Carbon-Agnostic`	9.92%	4.08%	13.00%	20.62%
`ST[35]`	11.92%	11.00%	9.92%	9.00%
`ST[10]`	6.54%	5.62%	8.54%	13.00%
`DT[40, 80]`	7.46%	11.38%	15.46%	15.08%
other algorithms	45.69%	45.54%	28.92%	27.54%

overall carbon consumption. We denote variations of this algorithm as $\texttt{DT}[L, H], s.t. H - L = \alpha$, where L and H represent different low and high threshold settings.

WaitAWhilePred: This algorithm utilizes predictions of both job length, denoted $\hat{c}$, and carbon intensity. Upon receiving these forecasts, it selects the $\lceil \hat{c} \rceil$ time slots with the lowest predicted carbon intensities within the given deadline T. The job is then scheduled to execute during these selected slots. The baseline version of this algorithm, which incorporates knowledge of the actual job length and carbon intensity forecasts, was proposed in [20].

Carbon-Agnostic: This algorithm employs a greedy approach that starts executing the job at the time of the submission and continues uninterrupted until completion, disregarding both current and future carbon intensities.

Finally, since the actual job length c is unknown to the above algorithms, and to ensure meeting the deadline constraint, algorithms will initiate *compulsory execution* when the remaining time slots are less than or equal to the time needed to run the remaining job length assuming its length is $c_{\texttt{max}}$.

3.2 Motivating `FTL`

The performance of carbon-aware scheduling algorithms is greatly influenced by factors such as job characteristics, switching overhead, job length, local environmental conditions (e.g., carbon intensity forecast accuracy), and an electricity grid's (i.e., a geographical region) idiosyncratic patterns. For instance, in a setting of jobs with high switching overhead, algorithms that neglect the switching overhead (e.g., `Single Threshold`, `WaitAWhilePred`) tend to perform poorly compared to `Double-Threshold` algorithm that makes pause/resume decisions based on substantial differences in carbon intensity. Moreover, job length significantly impacts threshold settings for both `Single Threshold` and `Double-Threshold` algorithms. Shorter jobs benefit from stricter thresholds, which minimize their carbon footprint during brief periods of very low carbon intensity. In contrast, longer jobs with stricter thresholds run the risk of requiring compulsory execution at the deadline during a high carbon intensity period. Thus, we parametrize the threshold values to facilitate a more effective carbon awareness.

Additionally, `WaitAWhilePred` excels when carbon intensity forecasts and job length prediction are accurate, especially when switching overhead is minimal. By greedily selecting low-carbon execution slots, it naturally recovers a nearly optimal solution. Conversely, the `Carbon-Agnostic` algorithm, which executes regardless of carbon intensity variations, is advantageous in environments with minimal changes in carbon intensity, significant forecast errors, or high switching overheads.

Table 1 evaluates the performance of the `Single Threshold` (`ST`), `Double-Threshold` (`DT`), `Carbon-Agnostic`, and `WaitAWhilePred` algorithms between 2020 and 2022 for jobs with medium switching overhead and a 10% error in both carbon intensity and job length forecasts (details of this experimental setup are explained in Section 4.1). The table reports the percentage of cases where an algorithm had the lowest carbon consumption. Focusing on the performance within any given region, it becomes apparent that no single algorithm consistently outperforms others. For example, in AU-NSW, the leading algorithm, `DT[40, 60]`, is only superior in ~25% of the instances. Moreover, the rankings of algorithms are not maintained across different regions, highlighting their sensitivity to local conditions. For instance, in Ontario, `DT[40, 60]` is the most effective, achieving 18.46%, whereas, in France, it drops in preference, with `Carbon-Agnostic` leading at 20.62%. We further detail the performance (carbon savings) implications of these algorithms in Section 4.2. This performance fluctuation calls for a solution that can *automatically* select the best algorithm for each job, ensuring both superior and consistent carbon emissions reductions.

3.3 `FTL`: Meta-Algorithm

To address this challenge of diverse performance orderings in Section 3.2, we propose `FTL`, a meta-algorithm inspired by the **F**ollow **t**he **L**eader strategy [9] that can match the performance of the best algorithm in each situation without requiring, e.g., hand tuning during deployment. The *meta-algorithm* we propose considers the historical performance of a full suite of algorithms to dynamically select the optimal approach for incoming jobs, considering their switching overhead and predicted job length. Figure 1 describes the flow of `FTL`. For each incoming batch job, `FTL` searches the database of various baseline algorithm variants, choosing the algorithm that has demonstrated the lowest carbon consumption for jobs with similar lengths. These comparable jobs are defined by lengths within the range $[\hat{c} - m, \hat{c} + m]$—where m is a margin indicating tolerance around the predicted length—and possess identical switching overhead characteristics. Subsequently, to refine its selection process and enhance future predictions, `FTL` simulates the remaining baseline algorithms against the current job to gauge their carbon consumption. This new data is then integrated into the database, ensuring that `FTL`'s decision-making continuously evolves and improves with each job, maintaining updated records of carbon consumption for all considered algorithms. This dynamic update mechanism ensures that `FTL` remains adaptive and precise in optimizing carbon savings across varying job types and settings.

`FTL` holds the potential to outperform traditional algorithms. Firstly, it removes the requirement for manual tuning of algorithm parameters by automatically adjusting to current conditions using historical data. Secondly, it can adapt to environmental changes, such as fluctuations in carbon intensity. This adaptability can be achieved by assigning greater weights in its decision-making process to the algorithms' carbon consumption for more recent jobs, ensuring that it remains responsive to recent environmental variations. Finally, it abstracts the relation between the job length, carbon intensity variations, and the effect of the algorithm by selecting the algorithm that behaved the best for similar cases.

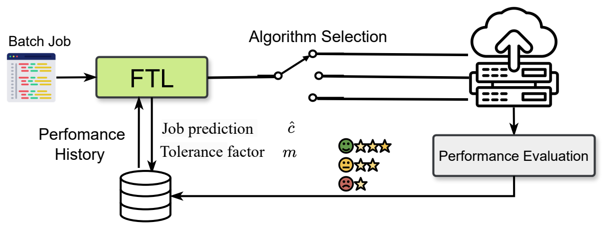


Figure 1: FTL Design

4 EXPERIMENTAL RESULTS

In this section, we experimentally evaluate FTL in reducing the carbon footprint of interruptible and delay-tolerant batch workloads.

4.1 Experimental Setup

Carbon intensity trace. We use carbon intensity traces from 2020 to 2022 for California, France, Australia-New South Wales (AU-NSW), Texas, and Ontario from ElectricityMaps [14]. The traces provide the hourly average intensity values, measured in gCO2eq/kWh. We select these regions to represent all quartiles of high and low average carbon intensity and high and low daily variations in carbon intensity, expressed as the coefficient of variation [19]. We assume that carbon intensity forecasts are error-prone, where we introduce uniform random errors to the data, denoted as CI_{err} that represents the mean percentage error added to the trace within the job's availability. We evaluated all algorithms against CI_{err} of [5%, 10%, 15%, 20%], which represents typical error reported by carbon intensity forecasts [12].

Job characteristics. We select 6500 job arrival times from 2020 to 2022, spaced 5 hours apart, to evaluate carbon footprint reduction across seasons and times of day. Each job arrives independently with a length, c, uniformly sampled within the range of [1, 24] hours. We assume that the actual job length is unknown to the scheduler; rather, an inaccurate job length prediction, denoted as $\hat{c}$, is provided. To incorporate the job length prediction error, we model a predictor that yields a job length estimate within the range $[c - J_{err} \times c, c + J_{err} \times c]$, where J_{err} is the percentage error in job length predictions. In addition, we assume that jobs are interruptible, where the job's state can be checkpointed and restored. We evaluate the performance by assuming that each job has a checkpoint/restore overhead of 1 minute (low), 5 minutes (medium), or 10 minutes (high), and each time unit is 1 hour. These durations represent the time required to complete a single checkpoint or resume operation. Finally, we evaluate each algorithm's carbon savings across different deadlines (T) for each job, including [24, 48, 72, 96] hours. Unless otherwise mentioned, we use a deadline of 48 hours.

Baseline Algorithms Configuration. To address the diverse characteristics of jobs, such as length and overhead, that significantly influence the performance of different scheduling algorithms, we construct a suite of baseline algorithms. These algorithms utilize predictions of carbon intensity till the deadline T.

(1) `Single Threshold`: We have developed a suite of Single Threshold (ST) algorithm variants, each defined by a threshold from percentiles [10, 15, 20, 25, 30, 35, 40] of predicted carbon intensity for the next T hours post-job submission. Each variant is labeled as $\texttt{ST}[p]$, where p represents the chosen percentile threshold.
(2) `Double Threshold`: We have a set of Double-Threshold(DT) algorithms denoted as $\texttt{DT}[L, H] (H > L)$, where H and L represent the high and low thresholds, respectively. The thresholds are selected from the predicted carbon intensity percentiles for the next T hours as the job is submitted, with $H \in \{20, 40, 60, 80\}$ and $L \in \{10, 20, 30, 40\}$ percentiles.

`WaitAWhilePred` and `Carbon-Agnostic` are also included as baseline algorithms. `WaitAWhilePred` leverages both job length prediction and carbon intensity forecasts to optimize scheduling, whereas `Carbon-Agnostic` adopts a strategy of continuous operation irrespective of carbon intensity predictions, focusing solely on job completion.

Evaluation Metric. We assess each algorithm's performance by comparing its carbon consumption to the `Carbon-Agnostic` algorithm. This is quantified using the carbon savings percentage (CS%): $\texttt{CS\%}_{\text{ALG}} = \left(\frac{\texttt{Agnostic-ALG}}{\texttt{Agnostic}}\right) \times 100$. This metric measures each algorithm's carbon efficiency relative to a baseline where carbon intensity is disregarded.

4.2 Evaluating FTL

The carbon footprint of scheduling algorithms can vary significantly based on regional characteristics, particularly the variability and average carbon intensity. Figure 2 illustrates the top five algorithms in each region that achieved the highest average carbon savings percentage across all jobs. In these comparisons, the error margins for carbon intensity (CI_{err}) and job length (J_{err}) predictions are set to 5% and 10%, respectively. As shown in Figure 2, our proposed FTL algorithm consistently outperforms other algorithms. For instance, in Ontario (Figure 2a), FTL achieves an 8% improvement over the best algorithm, ST[35], while in AU-NSW (Figure 2c), FTL obtain a 13% improvement over the best algorithm, DT [40, 60]. Finally, we note that across all regions, FTL consistently outperforms the closest runner-up, DT [40, 60], by 8.2%, achieving 14% carbon savings compared to running in a carbon-agnostic manner.

Figure 3 evaluates the algorithms' performance from a different perspective, focusing on the 10th percentile carbon savings rather than the average. The five algorithms with the highest savings at this percentile are investigated in each region. The negative savings observed for the algorithms are a result of their compulsory execution, due to the unknown actual job length. Interestingly, the algorithms' relative performance differs when comparing the average and 10th percentile metrics. For example, in Ontario, the Single Threshold algorithm, ST [35], is the closest runner-up based on average carbon savings (Figure 2a) but drops to fourth place when considering the 10th percentile carbon savings (Figure 3a). In contrast, FTL not only consistently achieves the highest average carbon savings across all regions but also exhibits robustness by consistently ranking in the top three algorithms for 10th percentile carbon savings. This demonstrates FTL's ability to perform well in both typical and more challenging scenarios.

Key Takeaways: *FTL outperforms the closest runner-up by 8.2% on average across all regions, resulting in 14% carbon savings compared to running in a carbon-agnostic fashion.*

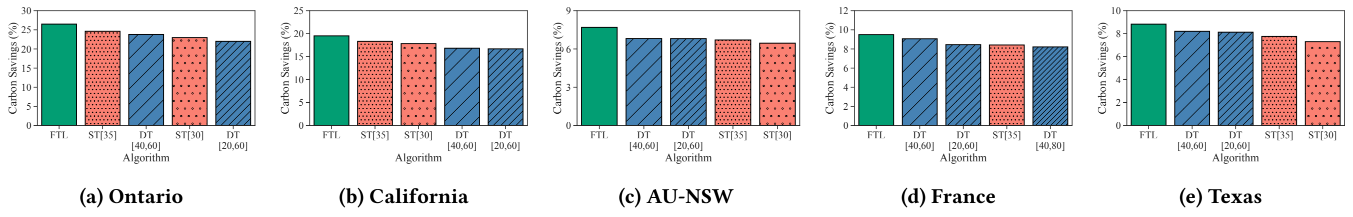


Figure 2: Algorithms' average carbon savings percentage with CI_{err} = 5%, J_{err} = 10%

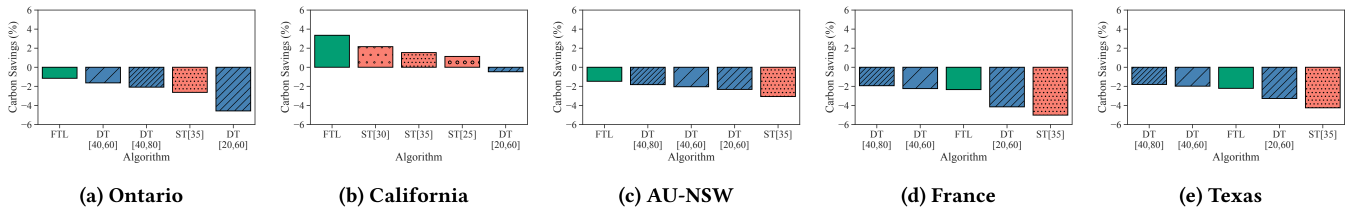
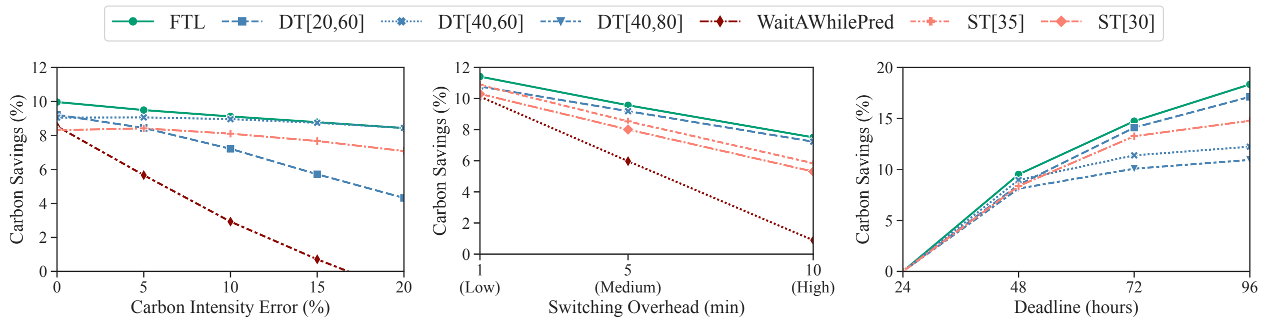


Figure 3: Algorithms' 10th percentile carbon savings percentage with CI_{err} = 5%, J_{err} = 10%



(a) Impact of Carbon Intensity Forecast, J_{err} = 10%, Deadline = 48hours

(b) Impact of Switching Overhead, CI_{err} = 5%, J_{err} = 10%, Deadline = 48hours

(c) Impact of Deadline, CI_{err} = 5%, J_{err} = 10%

Figure 4: Impact of different parameters on average carbon savings percentage in France

4.3 Evaluating Effect of Parameters on FTL

Effect of Carbon Intensity Forecast Error: The efficacy of algorithms in minimizing carbon emissions relies on the accuracy of future carbon intensity forecasts within the job's deadline. For threshold-based algorithms, these predictions inform the threshold settings, whereas `WaitAWhilePred` directly uses predictions to decide when to run or pause the job. However, as the margin of error in these predictions increases, so does the potential for suboptimal decision-making, leading to diminished carbon savings. In Figure 4a, we study the impact of CI_{err} on the average carbon savings of algorithms in France, with J_{err} = 10% over all jobs. We focus on the top five algorithms with the highest carbon savings at CI_{err} = 0%, extending the carbon forecast error to 20%. As shown, although average carbon savings decrease with increases in CI_{err}, `FTL` maintains its lead, underlining its robustness. In contrast, algorithms such as `WaitAWhilePred` and `DT[20, 60]` are significantly influenced.

Key Takeaways: *Rising errors in carbon intensity forecasts typically impair the performance of carbon-aware scheduling algorithms; however,* `FTL` *'s adaptive design continues to demonstrate superior carbon savings, showcasing its robustness against prediction uncertainties.*

Effect of Switching Overhead: The carbon efficiency of scheduling algorithms is markedly affected by overheads associated with job checkpointing and restoring. Algorithms that do not account for switching overhead tend to consume more carbon, especially when these overheads are significant. As the switching overhead increases, the carbon savings offered by all algorithms decrease due to the extra carbon expended during state transitions. In Figure 4b, we examine the influence of switching overhead in a setting with job length prediction error (J_{err}) set at 10% and carbon intensity forecast error (CI_{err}) at 5%, using France's carbon intensity trace. The figure presents the top five algorithms that had the highest carbon savings for low-overhead jobs and plots their average carbon savings percentage as the switching overhead increases. Notably, FTL maintains its lead, demonstrating robust carbon savings even with heightened switching overhead. Furthermore, the figure shows that as the overhead increases, the Double-Threshold algorithm, DT[40, 60], surpasses the Single Threshold, ST, and WaitAWhilePred algorithms.

Key Takeaways: *Switching overhead increase limits the possible carbon savings. Nonetheless,* FTL *maintains its superior performance.*

Effect of Deadline: As the deadline (T) increases, it provides the algorithms with higher flexibility to pause or resume and achieve higher carbon savings. In Figure 4c, we explore the impact of extending the deadline (T) from 24 to 96 hours in France, with $CI_{err} = 5\%$ and $J_{err} = 10\%$. We selected the top five algorithms that demonstrated the highest carbon savings at $T = 48$ hours. Notably, at $T = 24$ hours, all algorithms exhibit zero carbon savings akin to the Carbon-Agnostic approach. This occurs because, upon submission, algorithms are unaware of the actual job length. Thus, they assume that the maximum possible remaining job time ($c_{\max}$) is left, and since the deadline is also equal to $c_{\max}$ ($T = c_{\max} = 24$ hours), all the algorithms must start compulsory execution immediately. As anticipated, increasing the deadline leads to enhanced carbon savings across all deadlines, with FTL consistently achieving the highest savings.

Key Takeaways: *Increasing the deadline (T) allows for higher carbon savings. Again* FTL *consistently outperforms its peers.*

5 CONCLUSION

In conclusion, this paper presents FTL, a data-driven meta-algorithm that enhances carbon-aware scheduling in computing environments. Through extensive testing, we have shown that FTL outperforms traditional algorithms by adapting to real-time data and historical trends. By aligning job length predictions with historical performance, FTL achieves 8.2% and 14% average carbon reductions over the closest runner-up algorithm and carbon-agnostic scheduling across all regions, respectively. Future research will involve designing meta-algorithms for other scheduling modalities, such as scaling, where prediction errors highly impede carbon savings.

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, and CNS-2102963. Prashant Shenoy's research is supported by NSF grants 2105494, 2213636, and 23091241.

REFERENCES

[1] International Energy Agency. 2023. Data Centres and Data Transmission Networks. https://www.iea.org/energy-system/buildings/data-centres-and-data-transmission-networks

[2] Maria-Florina Balcan. 2020. Data-driven Algorithm Design. In *Beyond the Worst-Case Analysis of Algorithms*, Tim Roughgarden (Ed.). Cambridge University Press, Cambridge, Chapter 29.

[3] 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 4–7, 2024, Singapore, Singapore.* https://doi.org/10.1145/3632775.3661942

[4] Ran El-Yaniv, Amos Fiat, Richard M. Karp, and Gordon 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

[5] Rishi Gupta and Tim Roughgarden. 2020. Data-driven algorithm design. *Commun. ACM* 63, 6 (2020), 87–94.

[6] 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.* 7 pages. https://doi.org/10.1145/3604930.3605709

[7] 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, Article 57 (December 2023), 28 pages. https://doi.org/10.1145/3626788

[8] Walid A. Hanafy, Qianlin Liang, Noman Bashir, Abel Souza, David Irwin, and Prashant Shenoy. 2024. Going Green for Less Green: Optimizing the Cost of Reducing Cloud Carbon Emissions. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3 (ASPLOS '24), April 27-May 1, 2024, La Jolla, CA, USA.*

[9] Adam Kalai and Santosh Vempala. 2005. Efficient algorithms for online decision problems. *J. Comput. System Sci.* 71, 3 (2005), 291–307.

[10] 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).* https://doi.org/10.1145/3652963.3655074

[11] 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. https://doi.org/10.1145/3626776 arXiv:2303.17551 [cs.DS]

[12] 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).* 198–207. https://doi.org/10.1145/3563357.3564079

[13] Jens Malmodin, Nina Lövehagen, Pernilla Bergmark, and Dag Lundén. 2024. ICT sector electricity consumption and greenhouse gas emissions–2020 outcome. *Telecommunications Policy* (2024), 102701.

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

[15] Ana Radovanovic, Ross Koningstein, Ian Schneider, Bokan Chen, Alexandre Duarte, Binz Roy, Diyue Xiao, Maya Haridasan, Patrick Hung, Nick Care, et al. 2022. Carbon-Aware Computing for Datacenters. *IEEE Transactions on Power Systems* (2022).

[16] Prateek Sharma, Tian Guo, Xin He, David Irwin, and Prashant Shenoy. 2016. Flint: Batch-Interactive Data-Intensive Processing for Transient Servers. In *ACM European Conference on Computer Systems (EuroSys) (EuroSys '16).* London, United Kingdom, Article 6, 15 pages.

[17] 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.* 1–15.

[18] 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 *ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS).* 252–265.

[19] Thanathorn Sukprasert, Abel Souza, Noman Bashir, David Irwin, and Prashant Shenoy. 2024. On the Limitations of Carbon-Aware Temporal and Spatial Workload Shifting in the Cloud. In *Nineteenth European Conference on Computer Systems (EuroSys).* Athens, Greece.

[20] 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 (Middleware).*

[21] Ali Zeynali, Bo Sun, Mohammad Hajiesmaili, and Adam Wierman. 2021. Data-driven competitive algorithms for online knapsack and set cover. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 10833–10841.