Carbon-Aware Scheduling for Cyber-Physical Systems in the Edge-Cloud Continuum

Johan Kristiansson
Department of Computer Science, Electrical and Space Engineering
Luleå University of Technology
Luleå, Sweden
johan.kristiansson@ltu.se

Jerker Delsing
Department of Computer Science, Electrical and Space Engineering
Luleå University of Technology
Luleå, Sweden
jerker.delsing@ltu.se

Thomas Ohlson Timoudas
Department of Computer Science
RISE Research Institutes of Sweden
Stockholm, Sweden
thomas.ohlson.timoudas@ri.se

***Abstract*—Cyber-Physical Systems (CPS) powered by Artificial Intelligence (AI) have the potential to revolutionize industries by enabling advanced analytics and autonomous decision-making. To support resource-intensive applications, there is often a need to dynamically allocate additional compute resources. The Edge-Cloud Continuum enables allocation and deployment of workloads across platforms, including IoT devices, edge clusters, and cloud environments. However, the growing computational demands of these systems can unfortunately result in increased energy consumption and higher carbon emissions.**

This paper investigates the development of a carbon-aware scheduler for the Edge-Cloud Continuum, designed to optimize workload placement by balancing energy consumption, temporal variations in carbon intensity and resource availability. Key contributions of the paper include a spatiotemporal scheduling algorithm, a discrete-event simulator capable of replaying realistic workloads from the MIT SuperCloud dataset, and a comprehensive empirical evaluation.

Findings from the paper show substantial reductions in carbon emissions by prioritizing renewable energy sources and time-shifting workloads to periods of lower carbon intensity. However, when clusters operate under high utilization, time-shifting can inadvertently result in significantly higher emissions. In such scenarios, simpler greedy algorithms can be more effective.

I. Introduction

Cyber-Physical Systems (CPS) play an important role in modern society by seamlessly integrating computations with physical processes [1]. This integration enables real-time monitoring, automation, and decision-making across a wide range of industries. As these applications grow in complexity and scale, the demand for efficient and scalable computational platforms becomes increasingly important. The rapid advancement in AI further underscores the need for new computing platforms that can meet growing computational demands.

The Edge-Cloud Continuum [2] is a new paradigm in distributed computing that aims to unify edge and cloud computing. By dynamically distributing workloads, it becomes possible to optimize allocation of computational resources across platforms. For example, AI-driven CPS applications can process time-sensitive tasks on local edge servers and run resource-intensive tasks on remote cloud systems.

However, as the demand for computing continues to grow, so does energy consumption. Data centers are expected to account for a significant share of global energy use [3], thus contributing to increased greenhouse gas emissions and accelerating climate change. Developing technologies to reduce carbon emissions is therefore critical to reduce environmental impacts.

Lowering carbon emissions often correlates with energy efficiency and costs. For example, scheduling jobs in regions or at times when renewable energy sources, such as solar or wind power sources, are available can reduce both costs and environmental impact, as renewable energy is typically cheaper than fossil-fuel-based alternatives. Micro-grids CPS can optimize renewable energy usage by delaying task execution until sufficient renewable energy is available. Edge platforms can process data locally to maximize the use of renewable energy and offload execution to cloud resources in regions with lower carbon intensity, thereby minimizing reliance on carbon-intensive infrastructures.

This paper addresses the challenges of designing a scheduler to minimize the carbon emissions of workloads across an Edge-Cloud Continuum. Specifically, the paper proposes a time-shifting algorithm to dynamically allocate workloads across platforms and delay executions to periods of lower carbon intensity. The long-term vision is to integrate the proposed scheduler with Arrowhead [4] and ColonyOS [5], a meta-operating system designed to build computing continuums.

The paper is organized as follows: Section II presents related work. Section III briefly describes the Arrowhead framework, followed by a definition of the research problem in Section IV. Section V presents the proposed scheduling algorithm, which is evaluated in Section VI using real-world data and simulations. Finally, Section VII concludes the paper and discusses future research.

II. Related Work

Recent advances in carbon-aware computing have explored workload scheduling to reduce carbon emissions. For instance, *CASPER* [6] is a scheduling algorithm that optimizes workload placement across cloud regions by leveraging regional variations in carbon intensity while maintaining latency requirements. Temporal workload shifting [7]–[10] has also shown promise in reducing emissions by aligning computational tasks with periods of lower carbon intensity. Jagannadharao et al. [9] investigate time-shifting strategies for Large Language Model (LLM) training using a two-threshold mechanism to pause and resume tasks based on carbon emissions. Similarly, Wiesner et al. [8] propose a framework for temporal workload shifting in data centers, demonstrating the effectiveness of delaying tasks to periods of lower carbon intensity. However, these approaches are limited to suspending and resuming workloads within a single platform and do not address workload placement across clusters.

To incorporate spatial placement, Hanafy et al. [10] propose *GAIA*. This carbon-aware scheduler balances carbon reduction, performance, and cost by scheduling workloads across clusters in cloud-based HPC environments. *CarbonClipper* [11] introduces a learning-augmented algorithm for spatiotemporal online allocation with deadline constraints across data centers, taking both temporal and spatial considerations into account.

While previous studies have demonstrated the benefits of carbon-aware scheduling, this paper specifically addresses continuum environments, which consist of several connected heterogeneous compute environments that are often resource-constrained. For example, in some Industrial IoT (IIoT) use cases, most data processing occurs at edge nodes. However, sudden surges in computational demand may require offloading tasks to external cloud servers. In such scenarios, renewable energy-powered edge servers must operate alongside regional data centers with varying CO_2 intensities.

Many previous studies, e.g. [12], [13] focus on theoretical approaches or machine learning-driven frameworks to reduce carbon emissions or costs. Although these solutions have merits, they can sometimes be challenging to implement in real-world systems. In contrast, this paper adopts a practical implementation perspective, proposing a scheduling algorithm based on heuristic and straightforward rules.

The main contribution of this paper is a time-shifting algorithm that incorporates both temporal and spatial considerations for workload placement across heterogeneous platforms. Furthermore, the paper presents a simulator capable of replaying realistic workloads derived from the MIT SuperCloud dataset [14], providing a tool to evaluate scheduling algorithms in computing continuum environments. Unlike previous studies, e.g. [12], [15], which often view computing continuums as large-scale federated computing infrastructures, this paper adopts an application-centric perspective, focusing on enabling applications to seamlessly utilize diverse computational environments. The following section discusses the relationship to Arrowhead and the ColonyOS framework.

III. Arrowhead Framework

Arrowhead is a service-oriented framework [4] designed to develop and deploy IoT and CPS applications in industrial environments. It provides tools and services to seamlessly integrate devices, services, and systems. Central to the Arrowhead architecture is the concept of *local clouds* [16], which are self-contained environments where services can operate independently without relying on external clouds or systems. However, to support advanced analytical tasks or computationally heavy AI applications, there is often a need to dynamically allocate additional compute resources, which may exceed the capacity of local clouds.

There are two primary approaches to adding additional computing capacity to local clouds. The first approach is to export relevant data to an external environment for analytical processing. Using Arrowhead's existing secure local cloud interaction mechanisms, data can be securely transmitted to a cloud-based service. The second approach is to extend the local cloud by adding external computing resources. This can be achieved using trusted VPN tunnels to securely execute analytics on a trusted cloud service. Both of these methods enable real-time monitoring and visualization of the local cloud's carbon footprint and resource allocation optimization.

Another method of adding more computing capacity is to leverage ColonyOS, a framework designed to unify and manage distributed computing resources across platforms as part of a computing continuum. ColonyOS enables a grid-like computing network to form, where computing resources, whether local, edge, or cloud, can be seamlessly added and utilized. By integrating ColonyOS with Arrowhead it becomes possible to dynamically extend the computational capacity of local clouds by leveraging edge servers or external cloud environments whenever needed.

A detailed explanation of how to extend local clouds or integrate with ColonyOS is beyond the scope of this paper. Instead, this work focuses specifically on the development of a carbon-aware scheduler, which can be integrated with ColonyOS and Arrowhead. The next section presents a mathematical formulation of the problem which serves as the foundation for defining the proposed time-shifting algorithm.

IV. Problem Formulation

Let C represent a set of edge or cloud clusters, with c denoting an individual cluster in C. Each cluster c is characterized by a CO_2 intensity function $i_c(t)$, which provides the predicted carbon intensity of using the cluster at a given time t, measured in grams of CO_2 per kilowatt-hour (gCO_2/kWh).

Let P be a set of processes, with p representing an individual process in P. Each process p is defined by a power consumption function $e_p(t)$, measured in watts (W), which returns the power required to execute the process at time t. Furthermore, each process p has a deadline function $d(p)$, specifying the remaining time before it must start. The set of processes running on cluster c at time t is denoted $P_c(t)$, while $t_{\text{submitted}}(p)$, $t_{\text{start}}(p)$ and $t_{\text{end}}(p)$ represent the submission, start and end times of process p, respectively.

The carbon emissions of an individual process can then be calculated as shown in Equation (1):

$$E_p = \int_{t_{\text{start}}(p)}^{t_{\text{end}}(p)} e_p(t) \cdot i_c(t)\, dt \tag{1}$$

The cumulative carbon emissions E for all processes and clusters can be formulated as a multi-objective minimization problem, as shown in Equation (2):

$$\min E, \text{ where } E = \sum_{p \in P} E_p, \tag{2}$$

minimizing over all possible assignments of processes to clusters subject to the constraints:

$$t_{\text{start}}(p) \leq t_{\text{submitted}}(p) + d(p), \quad \forall p \in P \tag{3}$$

$$\sum_{p \in P_c(t)} r(p) \leq R(c), \quad \forall t \text{ and } c \in C \tag{4}$$

Here, $r(p)$ represents the resource requirements of a process p, and $R(c)$ represents the total resource capacity of a cluster c. The goal is to develop a scheduling algorithm that minimizes E, while ensuring all processes meet their respective deadlines and also respecting resource availability.

Several strategies can be used to reduce carbon emissions. One strategy could be to assign energy intensive processes to clusters with low CO_2 intensity. Another strategy could be to maximize utilization of low CO_2 clusters by dynamically balancing workloads over time. Finally, processes can be delayed to align their execution with periods when the predicted CO_2 intensity of clusters is lower, taking advantage of temporal fluctuations in carbon intensity. The following section discusses how to leverage all these strategies to optimize carbon efficiency.

V. Scheduling Algorithms

Given that many scheduling problems with multiple constraints (e.g., deadlines, resource availability, etc.) are *NP-hard*, heuristic-based approaches can be used to find approximate solutions within feasible computational limits. Note that heuristics trade precision and accuracy for computational efficiency to solve complex optimization problems using simpler rules.

A. Greedy Cluster Selection

A simple approach is to design a greedy scheduling algorithm that assigns processes to clusters based on their CO_2 intensity. Algorithm 1 describes an algorithm that assigns processes to the cluster with the lowest CO_2 intensity, The algorithm consists of two main functions: Schedule and HasFreeResource. The HasFreeResource function checks whether a specific cluster c has sufficient resources available. The Schedule function iterates through the clusters, select clusters with available resources, and then assigns a process p to the cluster with the lowest CO_2 intensity.

Algorithm 1 Greedy Cluster Selection

```
1: function SCHEDULE($p$, $C$)
2:     $C_{\text{available}} \leftarrow \{c \in C \mid \text{HASFREERESOURCES}(c, p)\}$
3:     if $C_{\text{available}} \neq \emptyset$ then
4:         Assign $p$ to $c_{\text{selected}} \leftarrow \arg\min_{c \in C_{\text{available}}} i_c(t_{\text{current}})$
5:     end if
6: end function

7: function HASFREERESOURCES($c, p$)
8:     return $r(p) + \sum_{\tilde{p} \in P_c(t_{\text{current}})} r(\tilde{p}) \leq R(c)$
9: end function
```

The main advantage of Algorithm 1 is simplicity. However, it does not take into account power consumption or deadlines of processes, which may result in suboptimal performance. Despite these limitations, the algorithm serves as a useful baseline for evaluating more complex scheduling algorithms.

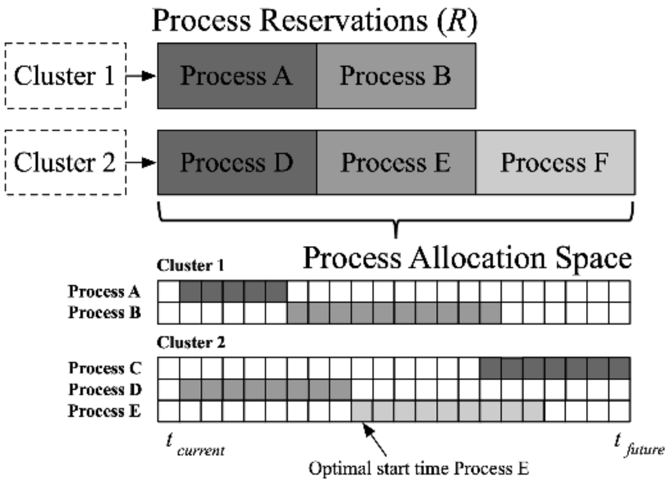


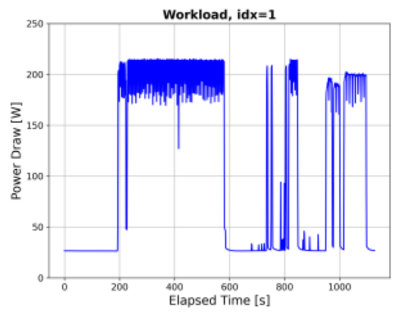
Fig. 1: Data structures used by time-shifting algorithm. Note that the allocation space may have overlapping processes if the cluster has many resources.

B. Time-shifting

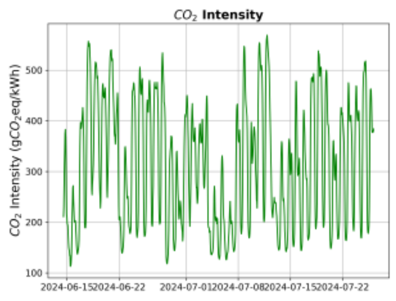
To leverage temporal variations in CO_2 intensity, it is necessary to delay execution of processes until the CO_2 intensity of the clusters reaches its lowest level. This approach requires predicting the CO_2 intensity of clusters over time and reserving time slots for running processes given their remaining deadlines before they to be started.

Figure 1 illustrates the data structures used by the proposed time-shifting algorithm. Each process is modeled as a data structure containing its predicted power usage, estimated execution time, planned start time, and assigned cluster. It is assumed that users specify a walltime, which defines the maximum duration a process can wait before it must run.

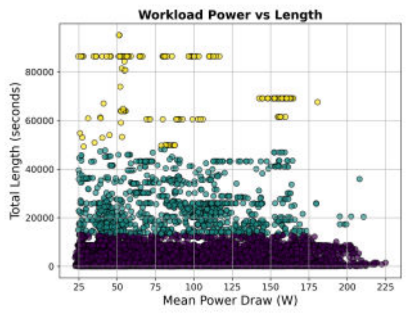
A hash table is used to store lists of reserved processes for each cluster. From these lists, it is possible to derive a matrix $\boldsymbol{A} : C \times T \rightarrow \mathbb{Z}$ representing the *Process Allocation Space*, where C is the set of clusters and T is the set of discrete times steps, which can be used to reserve and



(a) Workload example.



(b) Carbon intensity example.



(c) K-means clustering of the dataset.

Fig. 2: Workload examples.

determine available resources at any future time. The entry $\boldsymbol{A}(c,t) = \sum_{p \in P_c(t)} r(p)$ represents the amount of resources scheduled to be used on cluster c at the time instance t.

Algorithm 2 Time Shifting

```
1:  function SCHEDULE($p$, $\boldsymbol{A}$)
2:      $e_{\text{best}} \leftarrow \infty$, $c_{\text{best}} \leftarrow$ None, $t_{\text{best}} \leftarrow$ None
3:      for $c \in C$ do
4:          for $t \in [t_{\text{current}}, t_{\text{current}} + d(p)]$ do
5:              if HASFREERESOURCES($c, t, p, \boldsymbol{A}$) then
6:                  $e \leftarrow$ EMISSION(p, c, t)
7:                  if $e < e_{\text{best}}$ then
8:                      $e_{\text{best}} \leftarrow e$, $c_{\text{best}} \leftarrow c$, $t_{\text{best}} \leftarrow t$
9:                  end if
10:             end if
11:         end for
12:     end for
13:     if $c_{\text{best}} \neq$ None then
14:         $p$.planned_time $\leftarrow t_{\text{best}}$
15:         $p$.planned_cluster $\leftarrow c_{\text{best}}$
16:         # Update Process Allocation matrix
17:         for $t \in [t_{\text{best}}, t_{\text{best}} + p.\text{duration})$ do
18:             $\boldsymbol{A}[c_{\text{best}}][t] \leftarrow \boldsymbol{A}[c_{\text{best}}][t] + r(p)$
19:         end for
20:     end if
21: end function

22: function HASFREERESOURCES($c, t_{\text{start}}, p, \boldsymbol{A}$)
23:     for $t \in [t_{\text{start}}, t_{\text{start}} + p.\text{duration})$ do
24:         if $\boldsymbol{A}[c][t] + r(p) > R(c)$ then
25:             return false
26:         end if
27:     end for
28:     return true
29: end function

30: function EMISSION($p, c, t_{\text{start}}$)
31:     emission $\leftarrow 0$
32:     for $t \in [t_{\text{start}}, t_{\text{start}} + p.\text{duration})$ do
33:         emission $\leftarrow$ emission $+ e_p(t) \cdot i_c(t)$
34:     end for
35:     return emission
36: end function
```

Algorithm 2 outlines a method for assigning processes to clusters and determining an optimal start time by searching the Process Allocation Space. It calculates the estimated emissions for a process at different time steps to identify the scheduling option with the lowest emission. The HASFREERESOURCE function checks if a cluster has sufficient resources available at a specified time in the future. The function EMISSION estimates the carbon emissions of running a process at a given time. The SCHEDULE function finds consecutive time slots between the current time and the process's deadline that minimize the carbon emission for running the process.

VI. EVALUATION

To evaluate the performance of the proposed time-shifting scheduling algorithm, a discrete event simulator was developed to model workload execution. The simulator takes a dataset of time series workloads and a time series of CO_2 intensity as input and then replays the workloads across a specified set of clusters. Each cluster is associated with a CO_2 intensity log file and a predefined number of available GPUs, with GPUs serving as the only resource type for simplicity. The simulator can be configured to use different scheduling algorithms and outputs a dataset and statistical metrics generated from replaying the traces.

A. Dataset generation

The workload trace was extracted from the MIT SuperCloud dataset [14], which contains a time series of workloads that include power consumption. Note that the dataset does not contain the total energy consumption, but only the energy consumption of the GPUs collected using the *nvidia-smi* command. CO_2 intensity data for various clusters was obtained from Electricity Maps [17], which includes 24-hour predictions of CO_2 intensity.

For the experiments, a dataset of 10,000 workloads was randomly selected from the MIT SuperCloud dataset. Several log files were generated to record the start times of each workload. To control the rate at which workloads were replayed, a random waiting time between workloads was sampled from an exponential distribution, defined as waiting_time $\sim \text{Exp}\left(\frac{1}{\text{rate}}\right) \cdot$ scale, with rate $= 1$ and scale $= 100$ seconds. Both the CO_2 intensity and workload traces were re-sampled at a resolution of 1 second, meaning that the Process Allocation Spaces also had a 1-second resolution.

Figure 2 shows an example of the dataset[1], including a K-means clustering analysis that illustrates the relationship between power consumption and execution time. The K-means cluster reveals that the majority of workloads are short-lived and consume relatively small amounts of energy. However, a few workloads run for extended durations, reserving a GPU for long periods. This can be problematic if such workloads consume little energy, as they could prevent high-energy workloads from utilizing GPUs on clusters with lower CO_2 intensity. Additionally, it is quite common in the dataset for workloads to have low-energy period patterns followed by high-energy bursts. This means that the mean power usage might be relatively low, even though the GPU is fully utilized for extended durations. Figure 2a shows an example of such a workload.

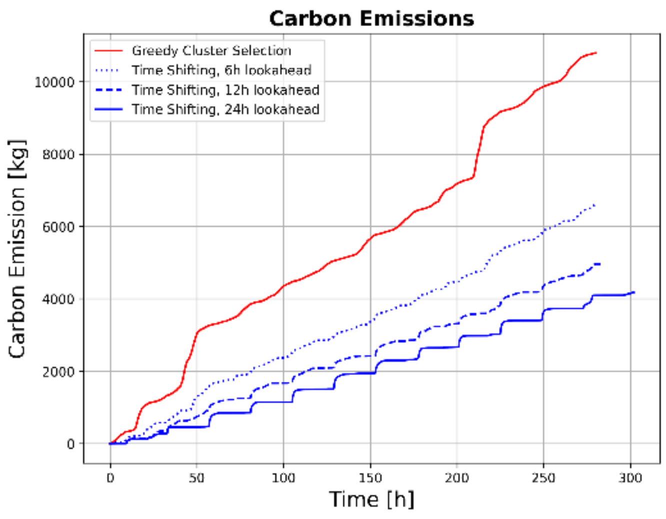


Fig. 3: Experiment 1 results.

1) Experiment 1 - Time-shifting: This experiment evaluated how effective the time-shifting algorithm was in reducing carbon emissions under no resource constraints. The simulator was configured to use a single cluster equipped with a large number of GPUs, ensuring that the cluster never ran out of resources. The CO_2 intensity dataset associated with the cluster was obtained from southern Sweden (SE-SE4). The results presented in Figure 3 show that the time-shifting algorithm outperformed the Greedy algorithm, reducing carbon emissions by 61%.

Figure 4 illustrates the CO_2 intensity over time and the carbon emissions of the time-shifting algorithm when process deadlines were set to 24 hours. As can be seen, the time-shifting algorithm effectively reduced emissions by leveraging periods of low CO_2 intensity, which aligns well with the expected behavior. However, when the process deadlines were reduced to 12 hours and 6 hours, the effectiveness of the time-shifting algorithm decreased, as shown in Figure 3.

[1]The dataset and the simulator will be made publicly available upon acceptance of the paper.

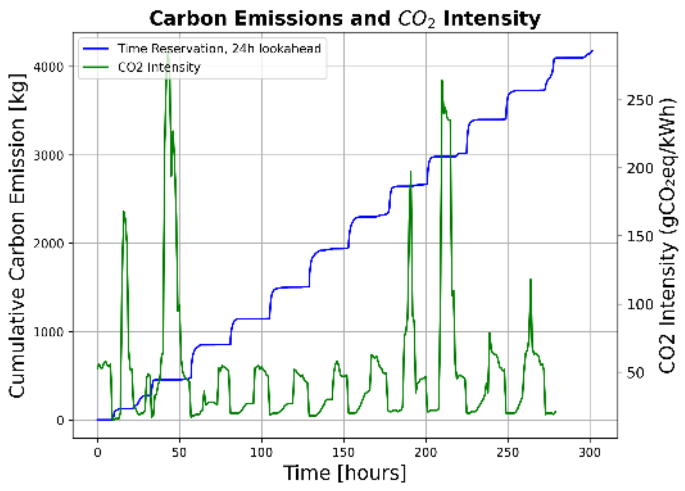


Fig. 4: Experiment 1 results - Time reservations and CO_2 intensity.

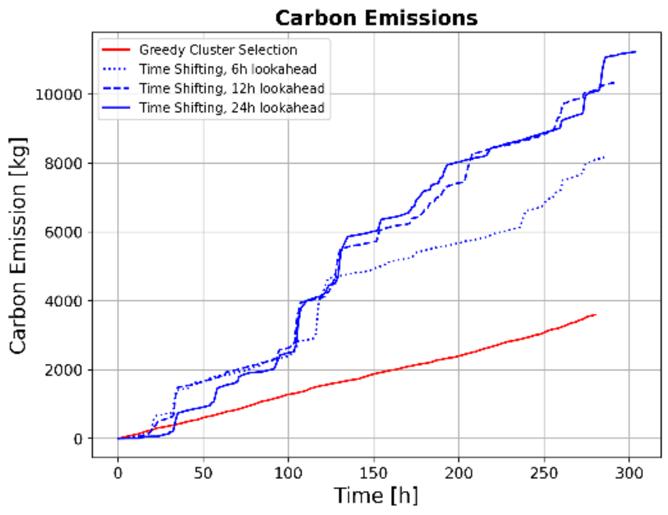


Fig. 5: Experiment 2 results.

2) Experiment 2 - High Utilization: This experiment aimed to evaluate the performance of the scheduling algorithms under conditions where the low CO_2 intensity clusters become fully utilized. The simulator was configured with two clusters. The first cluster, referred to as the green cluster, consisted of 32 GPUs located in northern Sweden (SE-SE1) and characterized by very low CO_2 intensity. The second cluster, referred to as the brown cluster, consisted of 127 GPUs, and was located in Poland, with significantly higher CO_2 intensity. This setup intentionally created an imbalance between green and brown compute resources and thus provided a controlled environment to assess how effectively the algorithms utilized renewable energy.

The results, presented in Figure 5, indicate that the time-shifting algorithm performed significantly worse than the simple Greedy algorithm. The reasons for this performance difference will be discussed in detail in the next section.

VII. Discussion

This paper proposes a carbon-aware scheduling algorithm for CPS applications operating within an Edge-Cloud Continuum. The experimental results demonstrate that time-shifting workloads can significantly reduce carbon emissions. However, when clusters become exhausted, time-shifting may instead lead to a substantial increase in carbon emissions.

A limitation of the time-shifting algorithm is its tendency to concentrate processes on clusters at certain start times. Although this approach reduced emissions in Experiment 1, it led to unbalanced resource utilization in Experiment 2. As more processes were reserved for future execution and available time slots became scarce, the algorithm's flexibility significantly decreased, forcing processes to be executed on the brown cluster.

In contrast, the Greedy algorithm demonstrated better performance by prioritizing the green cluster. One explanation for this is the randomized nature of the dataset and the fact that the Greedy algorithm naturally balanced long- and short-lived processes, which resulted in a more even workload distribution. A straightforward improvement to the time-shifting algorithm is to assign new processes to the latest available time slot rather than the optimal start time when cluster resources become exhausted, thus making the time-shifting algorithm perform as the Greedy algorithm when running out of resources and available time slots.

Handling processes that require immediate execution is another challenge. A simple solution is to over-provision the clusters to ensure sufficient capacity is always available. Alternatively, ongoing processes could be suspended to create room for new processes. Another potential solution could be to preemptively reschedule processes to other clusters or time slots to make room for urgent or high-energy processes. However, this approach would inevitably increase the algorithm's computational complexity, as it would be required to recursively evaluate the costs of reassigning already scheduled processes. Developing an efficient preemptive scheduling algorithm is an interesting topic for future research.

Another potential drawback of the time-shifting algorithm is that it can cause fragmentation in the Process Allocation Space. When a large number of short-lived processes are scheduled, the Process Allocation Space could become fragmented, making it challenging to allocate resources to long-lived processes. This phenomenon is somewhat similar to fragmentation in regular file systems. A potential solution to this problem could be to compress the Process Allocation Space. However, this approach could then result in suboptimal start times, thus reducing the benefits of time-shifting. More research is needed to explore the trade-offs of compressing the Process Allocation Space and the impact on carbon emissions.

To conclude, the paper has proposed a novel approach to carbon-aware scheduling for Edge-Cloud Continuums that could have significant implications for future CPS applications depending on compute-intensive AI workloads. In the future, we plan to integrate the scheduler with Arrowhead and ColonyOS, followed by real-world deployment and validation with real-world applications.

VIII. Acknowledgments

This work was supported by the European Commission through the Horizon Europe project SovereignEdge.Cognit under Grant Agreement 101092711, and the COP-Pilot Horizon Europe project under Grant Agreement 101189819.

References

[1] M. Achouch, M. Dimitrova, K. Ziane, S. Sattarpanah Karganroudi, R. Dhouib, H. Ibrahim, and M. Adda, "On Predictive Maintenance in Industry 4.0: Overview, Models, and Challenges," *Applied Sciences*, vol. 12, p. 8081, 08 2022.

[2] S. Moreschini, F. Pecorelli, X. Li, S. Naz, D. Hästbacka, and D. Taibi, "Cloud Continuum: The Definition," *IEEE Access*, vol. PP, pp. 1–1, 01 2022.

[3] Z. Cao, X. Zhou, H. Hu, Z. Wang, and Y. Wen, "Toward a Systematic Survey for Carbon Neutral Data Centers," *IEEE Communications Surveys and Tutorials*, vol. 24, no. 2, pp. 895–936, 2022.

[4] J. Delsing, Ed., *IoT Automation: Arrowhead Framework*. Boca Raton, FL: CRC Press, 2017.

[5] J. Kristiansson and T. Wikfeldt, "Developing AI Applications for the HPC-Cloud Continuum with ColonyOS," in *2024 23rd International Symposium on Parallel and Distributed Computing (ISPDC)*, 2024, pp. 1–8.

[6] A. Souza, S. Jasoria, B. Chakrabarty, A. Bridgwater, A. Lundberg, F. Skogh, A. Ali-Eldin, D. Irwin, and P. Shenoy, "CASPER: Carbon-Aware Scheduling and Provisioning for Distributed Web Services," in *Proceedings of the 14th International Green and Sustainable Computing Conference*, ser. IGSC '23. New York, NY, USA: Association for Computing Machinery, 2024, p. 67–73.

[7] E. Breukelman, S. Hall, G. Belgioioso, and F. Dörfler, "Carbon-Aware Computing in a Network of Data Centers: A Hierarchical Game-Theoretic Approach," in *2024 European Control Conference (ECC)*, 2024, pp. 798–803.

[8] P. Wiesner, I. Behnke, D. Scheinert, K. Gontarska, and L. Thamsen, "Let's Wait Awhile: How Temporal Workload Shifting Can Reduce Carbon Emissions in the Cloud," in *Middleware'21: 22nd International Middleware Conference*. ACM, 2021.

[9] A. Jagannadharao, N. Beckage, D. Nafus, and S. Chamberlin, "Timeshifting strategies for carbon-efficient long-running large language model training," *Innovations in Systems and Software Engineering*, pp. 1–15, 12 2023.

[10] W. A. Hanafy, Q. Liang, N. Bashir, A. Souza, D. Irwin, and P. Shenoy, "Going Green for Less Green: Optimizing the Cost of Reducing Cloud Carbon Emissions," *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, 2024.

[11] A. Lechowicz, N. Christianson, B. Sun, N. Bashir, M. Hajiesmaili, A. Wierman, and P. Shenoy, "CarbonClipper: Optimal Algorithms for Carbon-Aware Spatiotemporal Workload Management," 2024. [Online]. Available: https://arxiv.org/abs/2408.07831

[12] Z. Miao, L. Liu, H. Nan, W. Li, X. Pan, X. Yang, M. Yu, H. Chen, and Y. Zhao, "Energy and carbon-aware distributed machine learning tasks scheduling scheme for the multi-renewable energy-based edge-cloud continuum," *Science and Technology for Energy Transition*, vol. 79, 10 2024.

[13] Y. G. Kim, U. Gupta, A. McCrabb, Y.-B. Son, V. Bertacco, D. Brooks, and C.-J. Wu, "Greenscale: Carbon-aware systems for edge computing," *ArXiv*, vol. abs/2304.00404, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257912490

[14] S. S. et al., "The MIT SuperCloud Dataset," *CoRR*, vol. abs/2108.02037, 2021. [Online]. Available: https://arxiv.org/abs/2108.02037

[15] Y. Patel, P. Townend, A. Singh, and P.-O. Östberg, "Modeling the Green Cloud Continuum: integrating energy considerations into Cloud–Edge models," *Cluster Computing*, vol. 27, pp. 1–31, 04 2024.

[16] J. Delsing, J. Eliasson, J. van Deventer, H. Derhamy, and P. Varga, "Enabling IoT automation using local clouds," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, 2016, pp. 502–507.

[17] "Electricity Maps," https://www.electricitymaps.com, accessed: 2024-12-03.