Container Orchestration in Edge Computing with Fluctuating Green Energy: A Multi-Armed Bandit Approach

Václav Struhár[1], Alessandro V. Papadopoulos[1], Inmaculada Ayala[2,3], Mercedes Amor[2,3], and Lidia Fuentes[2,3]

[1] Mälardalen University, Sweden
[2] University of Malaga, Spain
[3] ITIS Software, Spain

Abstract. Sustainable edge computing demands intelligent scheduling of containerized workloads to exploit intermittently available renewable energy at geographically distributed sites. this work introduces a Contextual Multi-Armed Bandit (CMAB) framework for green-aware container orchestration, leveraging real-time context, such as energy availability, and resource utilization, via linear CMAB algorithms. A Python-based simulator models realistic solar and wind dynamics across regions. Compared to optimal, random, and naïve baselines, our CMAB scheduler improves green-energy utilization by up to 30% and cuts brown-energy use by 20%, while maintaining application performance guarantees. These findings underscore the potential of learning-based methods for advancing energy-efficient and sustainable edge infrastructures.

1 Introduction

Container orchestration platforms automate the deployment, scaling, and management of containerized applications, providing a flexible abstraction layer on diverse compute resources. In cloud data centers, orchestrators such as Kubernetes and Docker Swarm can rely on relatively homogeneous infrastructure and predictable workload patterns to achieve high resource utilization and service reliability. However, emerging applications (e.g., augmented reality, autonomous vehicles, environmental monitoring) require ultra-low latency and location-aware processing, which can not be met by cloud computing. The Edge Computing (EC) [15] paradigm takes advantage of the edge nodes placed at the Internet's frontier, proposing a more sustainable solution, while reducing the latency and resources demanded by the cloud. EC proposes to offload containerized services from the cloud onto nearby edge nodes/servers located in a range of one or two hops. This means that the data processing and storage of the IoT services and emerging applications can be moved to nodes located closer to where data and services are produced and consumed.

However, EC environments exhibit several unique characteristics that complicate orchestrators' placement decisions [17]: (a) Heterogeneity (Edge nodes vary

widely in CPU architecture, memory capacity, and software stack), (b) Workload Uncertainty (Application demands at the edge are highly dynamic, driven by user mobility, and event-driven spikes), and (c) Resource Contention (applications co-located on the same node compete for scarce resources).

Meanwhile, the growing environmental impact of large-scale cloud computing has spurred interest in integrating renewable energy into infrastructure planning. Data centers alone are projected to consume more than 10% of global electricity by 2025 [9, 2]. EC nodes offer new opportunities to utilize locally available green energy sources, such as solar panels or small-scale wind turbines [6], to power edge infrastructure. Orchestrator decisions should take into account green energy consumption, promote its use, and adopt differentiated scheduling strategies based on containers' resource demands and delay sensitivity, while prioritizing the use of green energy whenever possible. However, the intermittent and inherently unpredictable nature of renewable energy introduces additional scheduling complexity. Traditional heuristic or optimization-based schedulers struggle to adapt in real time to these dual uncertainties of workload and power supply. Forecasting inaccuracies can lead to missed opportunities for green energy utilization or, conversely, to degrade QoS when energy falls short of predictions. To navigate this trade-off, we turn to online learning methods that naturally balance exploration of underutilized resources against exploitation of high-yield opportunities.

This work formulates the container placement problem in edge environments as a Contextual Multi-Armed Bandit (CMAB) problem. Each edge node is modeled as an "arm", and at each decision epoch, the orchestrator observes a context vector comprising: (a) Real-time green energy fraction (ratio of renewable to total available power), (b) Current CPU, memory and I/O utilization, (c) Recent workload trends and QoS metrics (e.g., request latency, error rates). A CMAB algorithm uses these contexts to learn a scheduling policy that maximizes cumulative green energy usage, while ensuring that application performance remains within acceptable bounds. We validate our approach using a realistic simulation platform and demonstrate that our approach achieves up to 30% higher cumulative green energy utilization and 20% lower brown-energy consumption compared to state-of-the-art heuristics and non-contextual bandit baselines, without compromising QoS guarantees.

The main contributions of this paper are as follows:

1. **Formulation as Contextual Multi-Armed Bandit.** We rigorously formulate an energy-aware container placement in edge computing as an CMAB problem, capturing both resource states and renewable energy contexts in a unified framework.
2. **Adaptive Scheduling Algorithm.** We design and implement a scalable CMAB-based scheduling algorithm that dynamically prioritizes nodes with high renewable availability, yet continues to explore under-sampled nodes to adapt to shifting conditions.
3. **Realistic Simulation Platform.** We develop a Python-based simulator that integrates multiregion solar and wind trace datasets, heterogeneous

node capacities, and mixed-workload generation models, enabling reproducible evaluations under diverse scenarios.
4. **Comprehensive Evaluation.** Through extensive experiments, we demonstrate that our approach achieves up to 30% higher cumulative utilization of green energy and 20% lower brown energy consumption compared to state-of-the-art heuristics and non-contextual bandit baselines, without compromising QoS guarantees.

The remainder of the paper is organized as follows. Section 3 compares with some related work. Section 2 provides the necessary background. Section 4 formalizes our system and the problem model. Sections 5 and 6 detail the formal modeling and design of our scheduler. Section 7 describes the simulation environment and presents our experimental results. Finally, Section 8 concludes and outlines directions for future work.

2 Background: Multi-Armed Bandits

Multi-Armed Bandit (MAB) constitutes a straightforward but highly effective framework for algorithms that make sequential decisions in applications involving a degree of uncertainty [16]. Examples of applications of MAB are online advertising, clinical trials, website optimization, and finance.

An MAB algorithm has K possible actions to choose from, also known as arms, and T rounds. In each round, the algorithm chooses an arm and collects a reward for this arm. The reward is drawn independently from a fixed distribution (i.e., depends only on the chosen arm) but is unknown to the algorithm [16]. The algorithm receives feedback only for the selected arm after each round, while the rewards for the non-chosen arms remain unknown. Therefore, the algorithm must experiment with various arms through exploration to acquire new information. Thus, the algorithm needs a trade-off between exploration and exploitation: making optimal near-term decisions based on the available information. This trade-off arises in numerous application scenarios and is essential in MAB. The algorithm aims to learn which arms are best while minimizing the time spent exploring. There are several extensions of the MAB framework, e.g., the MAB has been extended in various directions to enable more complex decision-making scenarios. For example, the Contextual Multi-Armed Bandit bandit, which adds observable contextual information into the arm-selection process, and the combinatorial MAB, which enables the simultaneous selection of multiple arms.

3 Related Work

Sustainability in orchestration involves various metrics, such as brown and green energy consumption, as well as other environment-related indicators. Several works address this issue [7, 5, 8, 14]. In this work, we compare our proposed approach with existing studies that consider cloud-edge environments and aim to increase green energy usage or reduce carbon emissions. We have excluded works

focusing on Internet Energy or those centered on increasing the availability of green-powered nodes, as the complexity they introduce makes them not directly comparable to our approach.

KEIDS [10] is a scheduler with three objectives: maximize the use of green energy to minimize carbon emissions, optimize performance with minimal interference, and reduce overall energy consumption. This Multi-Objective Optimization problem is formulated as an Integer Linear Programming (ILP) problem that provides the optimal solution to map containers to pods. KEIDS shares with the approach presented here the multi-objective perspective (i.e., maximization of green energy, while maximizing performance) and energy model that considers edge nodes with different percentages of green energy. However, using ILP, which seeks the optimal solution, poses scalability problems.

Aldossary et al. [4] propose a multi-level approach using a mixed-integer linear programming (MILP) model to optimize application placement based on carbon intensity and traffic demands, aiming to minimize carbon emissions. The MILP optimization is combined with an offline heuristic that classifies nodes by their carbon emissions. Specific equations are used to calculate emissions for different network layers and fog/cloud nodes, integrating factors such as traffic demand and energy consumption. So, this work does not consider the contribution of green energy to reduce carbon emissions. In addition, it is a single-objective optimization process.

DECA [3] employs A* search algorithm and fuzzy sets to minimize energy costs and carbon emissions while ensuring sufficient capacity for application components. Its problem formulation includes resource availability constraints and considers intra- and inter-edge cloud communication costs and emissions. Like our approach, DECA must address conflicting goals. A key difference with the work presented here is that its energy model does not explicitly consider real-time fluctuations in energy source availability, such as those caused by intermittent renewable energy sources.

GreenFog [12] is a framework that operates in a Fog environment powered by on-site renewable energy and supplemented by grid energy when necessary. The framework dynamically adjusts QoS of IoT applications by scaling software containers and controlling tasks in IoT devices. GreenFog uses three strategies to optimize green energy utilization: exact optimal solutions using ILP, a heuristic approach based on linear regression, and an MAB approach. The paper concludes that the ILP approach achieves better results in terms of energy saving but requires precise knowledge of energy requirements and is computationally intensive, making it less practical for real-time applications. On the other hand, if properly trained, the MAB approach can have similar results. GreenFog seeks a trade-off between QoS (similar to performance in this approach) and using green energy as we do. However, its most effective optimization strategy (i.e., the one that uses Integer programming) introduces latency in scaling operations, such as shutting down nodes, which can take 10–12 minutes, and the MAB approach used that does not consider context requires extensive training to learn system patterns effectively.

The paper [13] proposes a novel scheduling approach that balances energy efficiency with QoS requirements, considering intermittent green energy sources and available batteries to supply energy. The optimization technique is a stable matching–based heuristic that utilizes distributed controllers that interact with a global controller in the context of a federated continuum. The adaptation action is to allocate serverless functions to nodes based on green energy availability and QoS constraints. The approach is compared with three baselines, random, first-fit, and local deployment, showing a potential to reduce energy consumption while maintaining system performance. So, the approach is not compared with other solutions with energy concerns.

As we can see, a majority of the approaches rely on exact methods like ILP, which have scalability problems and are impractical for real scenarios. Approaches like GreenFog [12] demonstrate that online learning methods have potential in this context. However, they require extensive training. In this context, we explore the use of Contextual Multi-Armed Bandits that can use contextual information to take more informed decisions.

4 System Model

This section presents a system model of edge computing with fluctuating green energy, as specified below.

Computing nodes: Edge computing consists of geographically distributed, heterogeneous nodes (n) that allocate limited resources (e.g., CPU (f_j^C), memory (f_j^M), and storage (f_j^S)) to containerized workloads. These nodes are networked and can communicate, but their energy sources vary by location. As a result, the availability of green energy differs across nodes based on local weather conditions. Each node is aware of the instant level of green energy.

Containers: A container (i-th container on node j π_j, i) is a lightweight, standalone, and executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries, and settings. Each container possesses specific resource requirements (i.e., CPU (π_k^C), memory (π_k^M), storage (π_k^S)). We assume that all nodes have pre-fetched copies of the container images; thus, the container image fetching time can be disregarded.

Green Energy: Each computing node operates on green energy (green energy at time t is denoted as $E(t)$), with the proportion of green to non-green energy fluctuating over time due to varying wind and solar intensities. We assume that it can be measured at any given time and expressed as a percentage range between 0% (no green energy) and 100% (powered purely by green energy).

5 Modeling

In CMAB, decision making is guided by additional contextual information that allows for more informed selections. A crucial aspect of modeling CMAB involves defining three fundamental elements: context representation, action space, and

reward function. The context representation captures relevant features of the system that influence the choice of actions. Action space defines options that the algorithm can take. The reward function quantifies the outcomes of actions.

Context Representation: Each context vector x_t captures the state of the system at time t and includes the following metrics for each node:

- Green Energy Availability ($e_{t,n}$): Measures the proportion or absolute value of green energy available at node n at time t.
- Schedulability Conditions ($s_{t,n}$): Reflects the ability of node n to meet the timing requirements of the container, possibly indicated by metrics like current task queue lengths, average response times, and missed deadlines.
- Computational Resources ($r_{t,n}$): Includes data on CPU utilization, memory usage, disk I/O, and network bandwidth for node n.

Therefore, the context for each node can be represented as a concatenated vector: $x_{t,n} = [e_{t,n}, s_{t,n}, r_{t,n}]$.

Action Space: The action at any decision point t involves selecting a node n from the set of all possible nodes N where the container could potentially be migrated. Thus, each action a_t directly corresponds to one specific node $a_t \in N$.

Reward Function: The reward function $R_t(a_t, x_t)$ evaluates the outcome of migrating a container to node n under the current context x_t. It is defined as a weighted sum of multiple criteria: $R_t(a_t, x_t) = \alpha f(e_{t,n}) + \beta g(s_{t,n}) + \gamma h(r_{t,n})$, where:

- $f(e_{t,n})$ is the energy efficiency reward that increases with higher green energy availability,
- $g(s_{t,n})$ is the schedulability reward that reflects the node's ability to meet timing guarantees,
- $h(r_{t,n})$ is the resource utilization reward that promotes efficient usage of computational resources,

and α, β, and γ are weights balancing the contribution of each term.

Contextual Multi-Armed Bandit Formulation: We formulate the container migration problem as a Contextual Multi-Armed Bandit problem. Each computing node $n \in N$ is modeled as an arm in the bandit framework. At each decision epoch t, the system observes a context vector $\phi(x_{t,n})$ for node n, which is a feature representation of $x_{t,n}$ (potentially including an intercept term). The expected reward from selecting node n is assumed to be a linear function of the context $r_{t,n} = \phi(x_{t,n})^\top \theta^* + \epsilon_{t,n}$, where $\theta^* \in \mathbb{R}^d$ is an unknown parameter vector and $\epsilon_{t,n}$ is a zero-mean noise term. The objective is to maximize the cumulative reward over a time horizon T: $R_T = \sum_{t=1}^{T} r_{t,a_t}$.

Feature Mapping ϕ for Container Migration: In the context of container migration, the raw context vector for each node is given by:

$$x_{t,n} = [e_{t,n},\ s_{t,n},\ r_{t,n}], \tag{1}$$

where:

- $e_{t,n}$ is the green energy availability at node n at time t,
- $s_{t,n}$ represents the schedulability conditions,
- $r_{t,n}$ denotes the computational resource metrics.

A feature mapping $\phi : \mathbb{R}^3 \to \mathbb{R}^d$ is applied to transform this raw context (containing green energy availability, schedulability conditions, and resource metrics) into a higher-dimensional feature space that is amenable to linear modeling. An effective choice is to include an intercept term and pairwise interaction terms among the metrics.

$$\begin{aligned}&\phi(x_{t,n}) = \\ &\left[1\ e_{t,n}\ s_{t,n}\ r_{t,n}\ e_{t,n} \cdot s_{t,n}\ e_{t,n} \cdot r_{t,n}\ s_{t,n} \cdot r_{t,n}\right]^\top.\end{aligned} \tag{2}$$

This mapping produces a 7-dimensional feature vector, enabling the model to capture both the individual contributions and the interactions between the metrics, which can significantly influence the effectiveness of container migration decisions. The feature representation ϕ is used in Algorithm 1 in Upper Confidence Bound Calculation and Reward Observation and Update.

5.1 Learning algorithm for optimal policy: LinUCB

Our approach employs the LinUCB algorithm [11] to balance exploration and exploitation by leveraging the linear relationship between the context and the reward. For each node n, the algorithm maintains $A_n = \lambda I \quad$ and $\quad b_n = \mathbf{0}$, where $\lambda > 0$ is a regularization parameter and I is the identity matrix. At each decision time t, for each node n, the following steps are executed:

1. **Context Observation:** Observe the context vector $x_{t,n}$ and construct its feature representation $\phi(x_{t,n})$.
2. **Parameter Estimation:** Compute the estimate of the parameter vector $\theta_n = A_n^{-1} b_n$.
3. **Upper Confidence Bound Calculation:** Compute the upper confidence bound (UCB) for node n:

$$p_t(n) = \phi(x_{t,n})^\top \theta_n + \alpha \sqrt{\phi(x_{t,n})^\top A_n^{-1} \phi(x_{t,n})}, \tag{3}$$

   where $\alpha > 0$ is a tunable parameter controlling the exploration-exploitation trade-off.
4. **Node Selection:** Select the node with the highest UCB:

$$n_t = \arg\max_{n \in N}\ p_t(n). \tag{4}$$

5. **Reward Observation and Update:** Migrate the container to node n_t and observe the reward r_t. Update the parameters for node n_t as follows:

$$A_{n_t} \leftarrow A_{n_t} + \phi(x_{t,n_t})\phi(x_{t,n_t})^\top, \qquad b_{n_t} \leftarrow b_{n_t} + r_t\, \phi(x_{t,n_t}). \tag{5}$$

The LinUCB algorithm's implementation for container migration in a fog-edge architecture is presented in Algorithm 1.

Algorithm 1 LinUCB for container migration decision

Initialize:
for each node n **do**
  $A_n \leftarrow \lambda I$, $b_n \leftarrow \mathbf{0}$
end for
for each decision point t **do**
  Observe context x_t for each node n
  Construct feature vector $\phi(x_{t,n})$ for each node n
  for each node n **do**
    $\theta_n \leftarrow A_n^{-1} b_n$
    $p_t(n) \leftarrow \theta_n^\top \phi(x_{t,n}) + \alpha\sqrt{\phi(x_{t,n})^\top A_n^{-1} \phi(x_{t,n})}$
  end for
  Execute migration to node n_t, Observe reward r_t
  Update: $A_{n_t} \leftarrow A_{n_t} + \phi(x_{t,n_t})\phi(x_{t,n_t})^\top$, $b_{n_t} \leftarrow b_{n_t} + r_t \phi(x_{t,n_t})$
end for

6 Orchestrator Architecture

In this section, we present the architecture of the orchestrator employing a CMAB. The orchestrator is the central component of the edge computing architecture as it drives overall system functionality and adaptability. Continuously gathers contextual information and uses this data to decide CMAB to migrate containers. The architecture of the orchestrator (see Figure 1) contains the following components:

- *Contextual Data Acquisition Module*to continuously gather contextual data from individual edge nodes.
- *Migration Evaluation Module* to continuously evaluate contextual data (such as green energy availability, schedulability, and resource status) to select candidate containers for migration.
- *CMAB* to perform the CMAB algorithm for each of the candidate containers for the migration (it decides where to migrate the container).
- *Constraints Checking Module* to ensure the feasibility of container migrations, compensating for the lack of constraint checks within the CMAB module.
- *Container Migration Manager* to trigger and manage the migration process in the edge computing environment.

7 Experimental evaluation

We developed a Python-based simulator for experimentation with CMAB-based and alternative orchestration strategies [1]. It supports infrastructure modeling, green energy dynamics, and policy tuning, enabling analysis of reward/regret behavior and scalable deployment scenarios.

This section evaluates the system's simulated performance under diverse scenarios, each defined by unique infrastructure setups and green energy patterns.

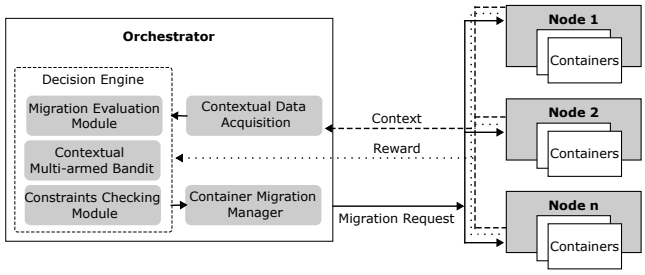


Fig. 1: Overview of the orchestrator.

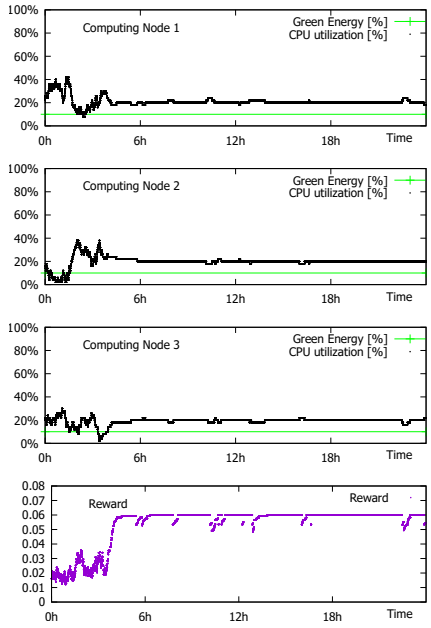
During a 24-hour period, nodes powered by varying renewable sources host container deployments, where placement decisions directly influence the cumulative reward of the system. Figures from 2a to 2c show the results of these experiments. In the figures, the top plots (denoted as Node 1, Node 2, ..., Node x) show the CPU utilization (shown as black lines or dots) and green energy levels (green lines). The higher the amount of containers deployed (thus the higher CPU utilization), the higher the lines (more extended deployment) or the dots (shorter deployment) are. The CPU utilization is expressed with values between 0% to 100%. Similarly, the green lines show the levels of green energy powering the respective nodes. After the node plots, the instant (R_t) reward is shown.

The first scenario (Figure 2a) shows three nodes with 30 containers in total. The green energy is constant and at the same level in all three nodes. After the initial exploration, the distribution of the containers evenly between all nodes, as the reward of CMAB depends on both green energy utilization and resource utilization given by the context.

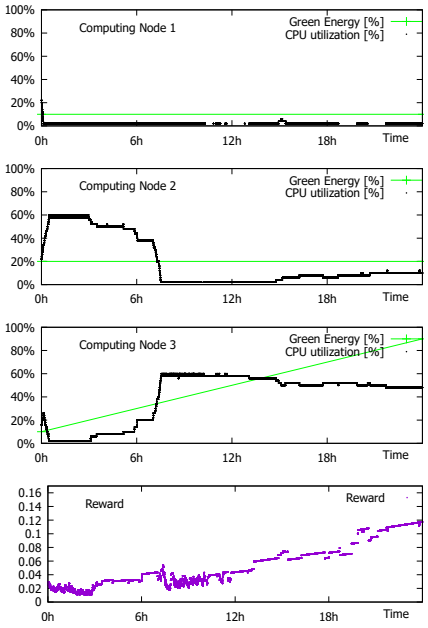
In the second scenario (Figure 2b), the green energy is constant in the first two nodes at 10% and 20%, respectively. The green energy at the third node increases linearly over time from 10% to 80%. So, we confirm the adaptability of the proposed method. At the beginning, the CMAB favors the second edge node as it offers the highest proportion of green energy. As the green energy in the third node surpasses the green energy in the second node (20%), the CMAB starts preferring the third node to migrate the containers to this node.

The scenario depicted in Figure 2c explores a more variable fluctuation in green energy. There are five nodes and 30 containers in total. There are spikes in green energy in nodes 1, 2, and 3. Node 4 has decreasing green energy, and node 5 has increasing green energy. We can see that the containers' deployment is aligned with the green energy levels. The most favorable is the first node, as it offers the highest proportion of green energy in the spikes. The system prefers the other available nodes when green energy ceases in the first node. Initially, node four is favored. Over time, as green energy availability increases in node 5, the preference gradually shifts toward node 5.

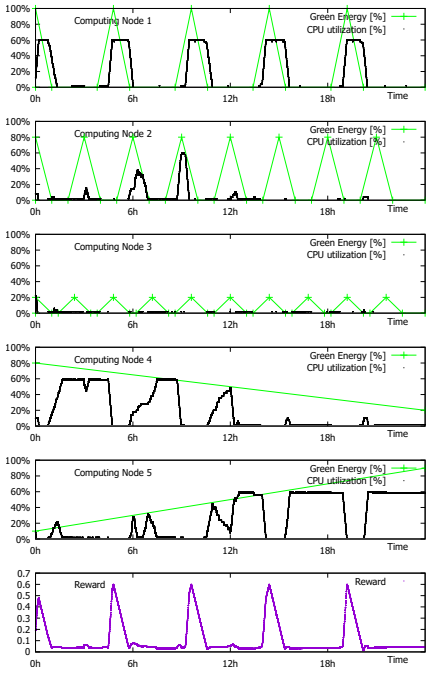
The previous experiments show artificially defined green energy levels to illustrate the behavior of CMAB. Here, we utilize the actual green energy distribution as recorded by Electricity Maps (`http://electricitymaps.com`). The Electricity Maps provides dataset records of hourly Carbon-free energy percent-



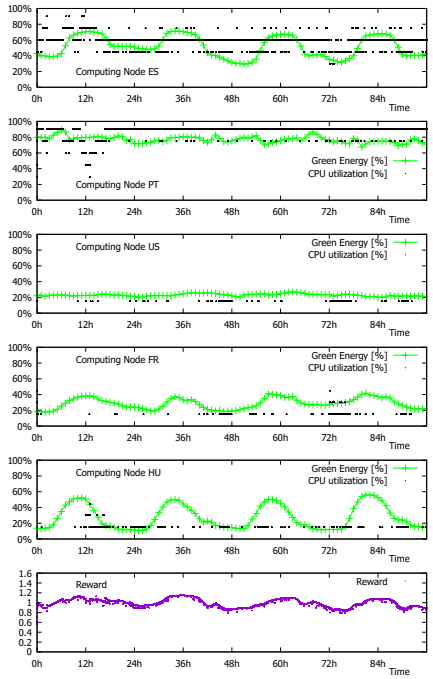
(a) Scenario 1: constant green energy.



(b) Scenario 2: linear green energy increase.



(c) Scenario 3: dynamic green energy.



(d) Scenario 4: green energy from Electricity Maps.

Fig. 2: Scenarios 1 to 4.

age. The setup consists of five nodes located in the United States, Portugal, Spain, France, and Hungary. The simulation spans over four days (August 20th-24th, 2024), during which four distinct increases in green energy occur due to solar availability in Spain and Hungary's edge nodes. The experiment shows the preference for deployment in the second node as it offers significantly more green energy. In this experiment, there are 10 containers, and each container requires 15% of CPU, thus they can not be deployed on a single node and need to be dispersed between more nodes. In this scenario, the first node is the second preferred one.

Lastly, we evaluated five different scenarios, comparing different deployment methods as shown in Figure 3. Each of the bars shows total reward; thus, the higher the bar, the better the deployment method in the specific scenario. The results show that CMAB performs comparably to the best deployment. We conducted this comparison in relatively small simulated infrastructures (ranging from 3 to 5 nodes with 5 containers) since an exhaustive search may not be feasible for larger infrastructures.

The conclusion of these experiments is that the CMAB approach is able to adapt to fluctuating green energy levels. Even for extremely fluctuating scenarios like Figure 2c, it is able to select the most beneficial node. For small scenarios, their results are nearly optimal.

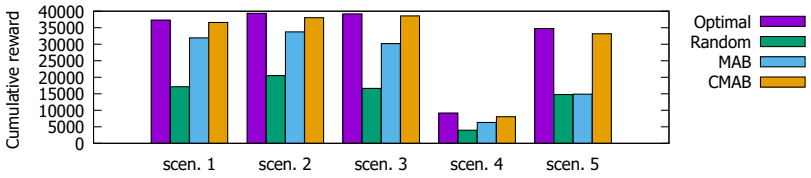


Fig. 3: Comparison of different decision-making policies.

8 Conclusion

This paper highlights the potential of Contextual Multi-Armed Bandit for optimizing energy-efficient scheduling in geographically distributed edge-computing systems. By prioritizing computing nodes with higher availability of fluctuating green energy, our approach enhances sustainability in resource allocation. Through a series of simulated experiments, we demonstrate the feasibility and effectiveness of this method compared to alternative policies, including the optimal policy, random selection, and a naive multi-armed bandit approach. The results underscore the advantages of our strategy in advancing energy-aware scheduling, contributing to the broader vision of greener infrastructures.

Acknowledgments. Work supported by the projects D2022-134337-T, *IRIS* PID2021-122812OB-I00, *SAVIA* PID2024-159945NB-I00 (co-financed by FEDER funds), and *DUNE* DGP_PIDI_2024_00092; and by ITIS/Universidad de Málaga.

References

1. Bandit simulation. `https://github.com/struharv/BanditsSimulation`, accessed: 2025-07-14
2. Ahmadisakha, S., Andrikopoulos, V.: Architecting for sustainability of and in the cloud: A systematic literature review. Information and Software Technology **171**, 107459 (2024)
3. Ahvar, E., Ahvar, S., Mann, Z.A., Crespi, N., Glitho, R., Garcia-Alfaro, J.: Deca: A dynamic energy cost and carbon emission-efficient application placement method for edge clouds. IEEE Access (2021)
4. Aldossary, M., Alharbi, H.A.: Towards a green approach for minimizing carbon emissions in fog-cloud architecture. IEEE Access (2021)
5. Bermejo, B., Juiz, C.: Improving cloud/edge sustainability through artificial intelligence: A systematic review. Journal of Parallel and Distributed Computing (2023)
6. Chen, X., Wen, H., Ni, W., Zhang, S., Wang, X., Xu, S., Pei, Q.: Distributed online optimization of edge computing with mixed power supply of renewable energy and smart grid. IEEE Transactions on Communications (2022)
7. Gaglianese, M., Soldani, J., Forti, S., Brogi, A.: Green orchestration of cloud-edge applications: State of the art and open challenges. In: IEEE International Conference on Service-Oriented System Engineering (SOSE) (2023)
8. Hao, Y., Cao, J., Wang, Q., Du, J.: Energy-aware scheduling in edge computing with a clustering method. Future Generation Computer Systems (2021)
9. He, W., Xu, Q., Liu, S., Wang, T., Wang, F., Wu, X., Wang, Y., Li, H.: Analysis on data center power supply system based on multiple renewable power configurations and multi-objective optimization. Renewable Energy (2024)
10. Kaur, K., Garg, S., Kaddoum, G., Ahmed, S.H., Atiquzzaman, M.: Keids: Kubernetes-based energy and interference driven scheduler for industrial iot in edge-cloud ecosystem. IEEE Internet of Things Journal (2020)
11. Li, L., Chu, W., Langford, J., Schapire, R.E.: A contextual-bandit approach to personalized news article recommendation. ACM, New York, NY, USA (2010)
12. N. Toosi, A., Agarwal, C., Mashayekhy, L., Moghaddam, S.K., Mahmud, R., Tari, Z.: Greenfog: A framework for sustainable fog computing. In: Service-Oriented Computing. Springer Nature Switzerland (2022)
13. Patel, Y.S., Townend, P.: A stable matching approach to energy efficient and sustainable serverless scheduling for the green cloud continuum. In: IEEE International Conference on Service-Oriented System Engineering (SOSE) (2024)
14. Perin, G., Meneghello, F., Carli, R., Schenato, L., Rossi, M.: Ease: Energy-aware job scheduling for vehicular edge networks with renewable energy resources. IEEE Transactions on Green Communications and Networking (2023)
15. Premsankar, G., Di Francesco, M., Taleb, T.: Edge computing for the Internet of Things: A case study. IEEE Internet of Things Journal (2018)
16. Slivkins, A.: Introduction to multi-armed bandits (2024)
17. Zhang, Y., Zhang, T., Zhang, G., Jacobsen, H.A.: Lifting the fog of uncertainties: Dynamic resource orchestration for the containerized cloud. In: Proc. of ACM Symposium on Cloud Computing (2023)