Carbon-Neutral Computing at the Edge: Scalable Greedy–LP Optimization of Green Data Centers

Kazuki Okazawa[1], Hiroki Nishikawa[1], Dafang Zhao[1], Ittetsu Taniguchi[1],
Takao Onoye[1], Marcos De Melo da Silva[2], and Abdoulaye Gamatie[3]
1 *Graduate School of Information Science and Technology, The University of Osaka, Osaka, Japan*
2 *Independent Researcher, Montpellier, France*
3 *LIRMM, University of Montpellier - CNRS, Montpellier, France*
email: okazawa.kazuki@ist.osaka-u.ac.jp

***Abstract*—As energy demands in cloud computing surge, efficient task scheduling and carbon-free energy use in mini data centers is essential. Solving scheduling problems with mixed-integer linear programming optimization methods is computationally intensive and less scalable. This paper proposes a two-step approach combining a greedy algorithm with linear programming to optimize virtual machines scheduling and energy management. We show via simulations that our method preserves solution quality, while accelerating the calculation process by 99.6%. While the literature method requires 6 hours to solve a reference system design, our solution handles 18 times larger designs in 20 minutes only, demonstrating its scalability.**

***Index Terms*—Edge data center, distributed computing, carbon neutrality, renewable energy, resource allocation, optimization**

I. Introduction

Over the past decade, the energy demand in data centers has surged due to the rapid growth of cloud computing and large-scale applications. Data centers now consume 200 TWh of electricity annually, accounting for about 1% of global electricity usage and 0.3% of global CO2 emissions [1], [2]. To address the demand sustainably, companies such as Google, Amazon, and Facebook are pursuing carbon neutrality by purchasing renewable energy credits [3]. While these credits support renewable energy production, they do not ensure direct utilization by data centers. For this limitation, Google has committed to matching its energy consumption with 24/7 renewable generation [4], while Facebook and Amazon aim for 100% renewable energy by 2025 and net-zero carbon emissions by 2040 [5], [6]. Despite these efforts, optimizing energy usage in data centers remains a significant issue [7].

Data center energy optimization in the literature: To improve energy efficiency, recent works have proposed optimization techniques for virtual machine (VM) placement based on communication patterns and workload demands, achieving notable energy savings [8]–[11]. Furthermore, optimization efforts that leverage renewable energy sources have been explored [12], [13]. These approaches involve deploying batteries and renewable sources at locations separate from data centers and optimizing VM allocation to utilize stored renewable energy. However, relying solely on VM placement for minimizing grid electricity has limitations. More advanced approaches integrate mini data centers equipped with batteries and solar panels, enabling the migration of both energy and VMs to maximize renewable energy utilization [14]. While effective, these methods often encounter scalability challenges due to the computational complexity of co-optimizing VM and energy migrations.

Challenges in VM migration optimization: VM migration optimization has long been a focus of research due to its impact on energy efficiency and system performance. Traditional greedy methods, such as First-Fit Decreasing (FFD) [15], Best-Fit Decreasing (BFD) [16], and Worst-Fit Decreasing (WFD) [17], have been proposed to optimize VM allocation. Meta-heuristic approaches, including ant colony optimization [18], genetic algorithms [19] and flower-pollination-based nondominant sorting optimization [20], and deep learning techniques [21] [22] are being explored to tackle multi-objective and complex problems. These methods aim to improve energy efficiency in data centers by optimizing the VM allocation. However, unlike [14], most of these methods do not consider renewable energy integration in VM migration processes, leaving a gap in achieving sustainable data center operations. Additionally, our preliminary experiments of [14] reveal that the vast number of possible VM migration significantly increases computational costs, highlighting scalability as a persistent issue.

Our contribution: To address these challenges, we propose a two-step approach that combines a greedy algorithm for VM migration with linear programming (LP) for energy allocation. First, the greedy algorithm analyzes compute node resources and projected CPU utilization to determine VM migration. Second, LP optimally allocates energy based on the VM allocation, ensuring scalability and energy efficiency. This approach reduces computational complexity while supporting the sustainable growth of data centers by efficiently utilizing renewable energy. Our contributions are as follows:

- Development of a two-step optimization approach for efficient VM and energy migration to reduce computational complexity of allocation solution finding.
- Experimental results demonstrating up to 99.6% reduction in computational time while preserving energy efficiency, enabling scenarios 18 times larger to be handled within realistic time frames, compared with [14].

Organization: The remainder of this paper is organized as

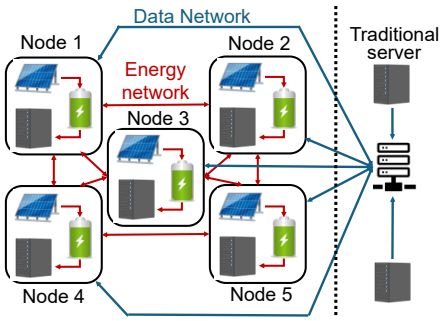


Fig. 1: System model.

follows. Section II outlines the system model and problem formulation of the existing work proposed in [14]. It also discusses drawbacks and insights through preliminary experiments. Section III introduces our two-step solution. Section IV presents simulation results and a detailed analysis of our solution costs. Section V finally concludes the paper with future remarks.

II. Existing MILP-based Allocation Framework

A. System Model

The system assumed in this paper is based on [14], which is designed to achieve energy neutrality. It consists of multiple green nodes ($N = \{1, \ldots, n\}$, e.g., $n = 5$ in Fig. 1), each equipped with a server, a photovoltaic (PV) panel, a battery for energy storage, and a logic board to manage energy generation, storage, and transfer. Energy generated by the PV panels (G_i^t) is stored in the battery without any charge or discharge loss and is used to power the server. Nodes are interconnected through a network, allowing both energy transfer and VM migration.

Energy transfer incurs losses determined by power electronics components like DC/DC converters and wires, with efficiencies depending on the path. For example, a direct path between two nodes achieves 85.8% efficiency, while an indirect path involving one intermediary node drops to 84.3%, with an additional 1.8% efficiency loss per extra intermediary. VM migration results in energy overheads at both the source (μ_s) and destination (μ_d) nodes, proportional to the data size.

Each node has resources including CPU capacity (RC_i), RAM (RM_i), disk (RD_i), and network bandwidth (RB_i). In this study, simplified models are adopted for all key components — computing, PV generation, and battery storage — in order to focus on the decision-making policies for VM and energy migrations rather than the hardware-level modeling of each resource. For computing, CPU capacity is represented not in MIPS (million instructions per second) as in [23], [24], but as a utilization ratio in [0.0,1.0], where 0.0 means idle and 1.0 indicates 100% usage, reflecting overall usage rather than per-core details. The adoption of a CPU-centric model follows the baseline in [14] to ensure a fair comparison, and aligns with the observation that CPU utilization is generally the most directly correlated with server power consumption in real-world data centers.

TABLE I: Variables and Parameters in the MILP Formulation.

Sets	**Description**
N, H, J	resp. num. of green nodes & VMs, and planning horizon
Parameters	**Description**
G_i^t	PV generation on node i at time t
$E_{i,k}$	Energy transfer efficiency between nodes i, k
C_j^t	CPU utilization of VM j at time t
R_i^C	CPU capacity resources on node i
L_i	Safety discharge level of battery in node i
U_i	Maximum capacity of battery in node i
I_i	Initial amount of energy stored in node i
ϵ_s	Static energy consumption
ϵ_p	Additional energy consumption when nodes at 100%
λ	Energy loss for transferring 1 Wh between nodes
μ	Total energy cost for migrating a VM
μ_s, μ_d	Source & Target server energy costs for migrating a VM
ϕ	Energy loss for injecting 1 Wh in the utility grid
ν	Penalization for server CPU overloading

Similarly, PV panels are characterized solely by their aggregate area and conversion efficiency, without accounting for temperature effects or partial shading. Batteries are modeled by total capacity and charge/discharge efficiency, excluding degradation and state-of-health dynamics.

Although such simplifications omit factors such as memory-related power consumption, network-induced energy usage, PV output variability, or battery aging, they allow the optimization problem to remain tractable while capturing the dominant factors influencing migration decisions. The proposed framework can be extended to incorporate these aspects in future work if a more comprehensive model is required.

The computational workload consists of m VMs ($J = \{1, \ldots, m\}$), each with requirements for memory, disk, bandwidth (V_j^M, V_j^D, V_j^B), and a time-varying CPU load (C_j^t). For simplicity, we do not explicitly focus on the allocation of other computational resources such as memory, disk and network bandwidth since we are only interested in modeling the energy consumption due to CPU utilization. The system's primary goal is to minimize the purchased brown energy (due to possible renewable energy shortage) while optimizing resource usage and energy transfers for VM execution.

B. Mixed-Integer Linear Programming (MILP) Formulation

To represent the inherent dynamic behavior of the system, we define a planning time horizon H, discretized into T time steps. Each time step corresponds to an interval of τ minutes.

Decision variables: We consider the following decision variables to find the energy and VM migrations:

- $x_{i,j}^t$: $0-1$ variable, which becomes one when VM j is running on node i at time t, otherwise 0.
- $z_{i,k,j}^t$: $0-1$ variable, which becomes one when VM j is migrated from node i to k at time t, otherwise 0.
- $v_i^t \geq 0$: overloaded CPU capacity on node i at time t
- $L_i \leq w_i^t \leq U_i$: battery energy level of node i at time t.
- $f_{i,k}^t \geq 0$: amount of energy migrated from node i to node k at time t.
- $b_i^t \geq 0$: amount of purchased energy on node i at time t.

- $q_i^t \geq 0$: amount of energy injected into grid by node i at time t.

Objective function and constraints: We formulate the considered objective function, as follows:

$$\begin{aligned}\text{Minimize } \lambda \sum_{t\in H}\sum_{i,k\in N} f_{i,k}^t + \mu \sum_{i,k\in N}\sum_{j\in J}\sum_{t\in H} z_{i,k,j}^t \\ + \sum_{t\in H}\sum_{i\in N} b_i^t + \phi \sum_{t\in H}\sum_{i\in N} q_i^t + \nu \sum_{i\in N}\sum_{t\in H} v_i^t\end{aligned} \tag{1}$$

which is subject to the following constraints:

$$\sum_{i\in N} x_{i,j}^t = 1, \forall t \in H, j \in J \tag{2}$$

$$z_{i,k,j}^t \geq x_{i,j}^t + x_{k,j}^{t-1} - 1, \forall t \geq 2, j \in J, i \neq k \in N \tag{3}$$

$$\sum_{j\in J} C_j^t x_{i,j}^t \leq R_i^C + v_i^t, \forall t \in H, i \in N \tag{4}$$

$$\begin{aligned}w_i^t =& w_i^{t-1} + b_i^t + G_i^t + \sum_{k\in N} E_{k,i} f_{k,i}^t \\ &- \sum_{k\neq i\in N} f_{i,k}^t - \mu_s \sum_{k\in N}\sum_{j\in J} z_{i,k,j}^t - \mu_d \sum_{k\in N}\sum_{j\in J} z_{k,i,j}^t \\ &- (\epsilon_s + \epsilon_p \sum_{j\in J} C_j^t x_{i,j}^t) - q_i^t, \forall t \in H, i \in N\end{aligned} \tag{5}$$

$$L_i \leq w_i^t \leq U_i, \forall t \in H, i \in N \tag{6}$$

$$w_i^0 = I_i, \forall i \in N \tag{7}$$

The above objective function aims to maximize overall energy efficiency within the system by minimizing energy losses due to inter-node energy and VM migrations. Additionally, it seeks to limit the amount of energy purchased from the utility grid and to minimize losses when excess energy is injected back into the grid. Finally, by reducing penalties for CPU overloading on each server, the function helps maintain system stability while optimizing energy management as a whole.

Constraints (2)–(4) govern VM assignments: (2) ensures each VM j is assigned to node i, (3) tracks VM migrations between time steps, and (4) limits total CPU utilization per node to server capacity. Constraints (5)–(7) handle state of charge (SoC): (5) models SoC changes, accounting for energy sources, transfers, workload, and migration costs; (6) enforces battery capacity and discharge limits; and (7) defines initial SoC.

C. Preliminary Experiments and Observations

Scalability bottleneck: To identify the bottleneck in this optimization problem, the resolution process is divided into two main parts: VM migration then energy transfer optimizations. Fig. 2a is based on our reproduction of the work in [14]. It compares the computational times of the two parts, as the number of green nodes increases. Since VM migration optimization is formulated as MILP problem, Fig. 2a clearly shows that its computational time increases quite fast.

Characteristics of VM behavior: Fig. 2b, 2c, and 2d illustrate the fluctuation of CPU utilization by VMs over the planning horizon. Fig. 2b shows VM activity with low standard deviation, while Fig. 2c illustrates a high standard deviation. Fig. 2d summarizes the mean and standard deviation for considered VMs. High fluctuation in CPU utilization often leads to more VM migrations, increasing corresponding energy losses. To address this, we propose a two-step approach and a suitable algorithm that determines VM migration reducing computation time while maintaining energy efficiency.

III. Proposed Scalable Two-step Approach

This section presents our two-step approach combining a greedy algorithm for VM migration with LP for energy allocation, reducing computational time and improving scalability. Unlike previous state-of-the-art MILP-based method [14], our approach achieves significant time savings. It is summarized in Fig. 3. The left-hand part of the figure shows the first step of our solution, i.e., VM migration scheduling, while the right-hand part focuses on energy migration scheduling, given with the VM migrations identified in the first step.

A. First-step Optimization based on Our Greedy Algorithm

The VM migration workflow, shown on the left part of Fig. 3, is detailed in Algorithm 1. The algorithm iterates through each node to identify overloaded servers (*source servers*), i.e., where CPU utilization (C_s) exceeds CPU capacity (R_s) (line 1-3). For each overloaded server, it initializes the migration cost (v) and the minimum migration cost (v_{min}) to explore the optimal VM combination to be migrated with minimum cost. (line 5).

To identify viable target servers, nodes are sorted by a weight factor (w) that prioritizes servers with sufficient remaining capacity ($R_d - C_d$), larger solar panel area (S_d), and higher battery capacity (U_d) (line 7). The weight factor is calculated as follows:

$$w = \{R_d - C_d\} \cdot \frac{S_d \cdot U_d}{S_{\max} \cdot U_{\max}}$$

where $S_{\max}$ and $U_{\max}$ denote the maximum solar panel area and battery capacity, respectively, across all servers.

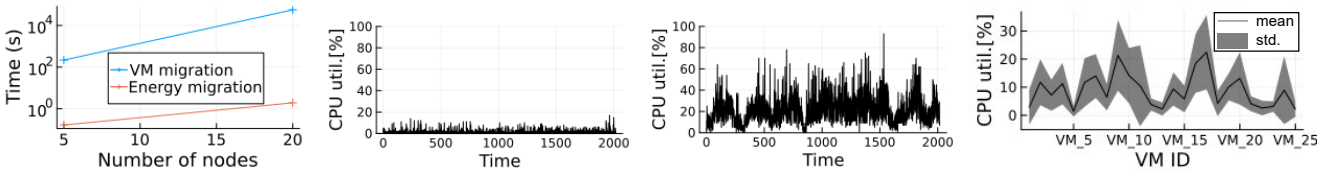
Next, the algorithm proceeds to the VM migration exploration phase (line 9), evaluating combinations of VMs to minimize the migration cost (v), calculated as follows:

$$v = \sigma_m J_m + P_e$$

where σ_m represents the standard deviation of the future CPU utilization for a VM, J_m is the number of VMs to be migrated, and P_e is a penalty for CPU overloading (line 10) defined as:

$$P_e = \begin{cases} 0 & \text{if } C_s \leq R_s \text{ and } C_d \leq R_d, \\ \nu \cdot (C_d - R_d) & \text{if } C_s \leq R_s \text{ and } C_d > R_d, \\ \nu \cdot (C_s - R_s) & \text{if } C_s > R_s \text{ and } C_d \leq R_d, \\ \nu \cdot (C_d - R_d + C_s - R_s) & \text{if } C_s > R_s \text{ and } C_d > R_d. \end{cases}$$

If a combination yields a cost (v) lower than the current minimum value (v_{min}) , the algorithm updates the optimal migration plan (line 14). If a combination meets the penalty threshold (P_e), it stops exploring further target servers to avoid unnecessary checks (line 16). After finding the optimal VM



(a) (b) (c) (d)

Fig. 2: Preliminary experiments to analyze the reproduced work in [14] as the state-of-the-art: (a) shows computational time of VM and energy migration, (b) and (c) present changes in CPU utilization of VM_5 (low std.) and VM_9 (high std.), respectively. Finally, (d) indicates the statistical analysis for each CPU utilization of all VMs for entire time horizon.

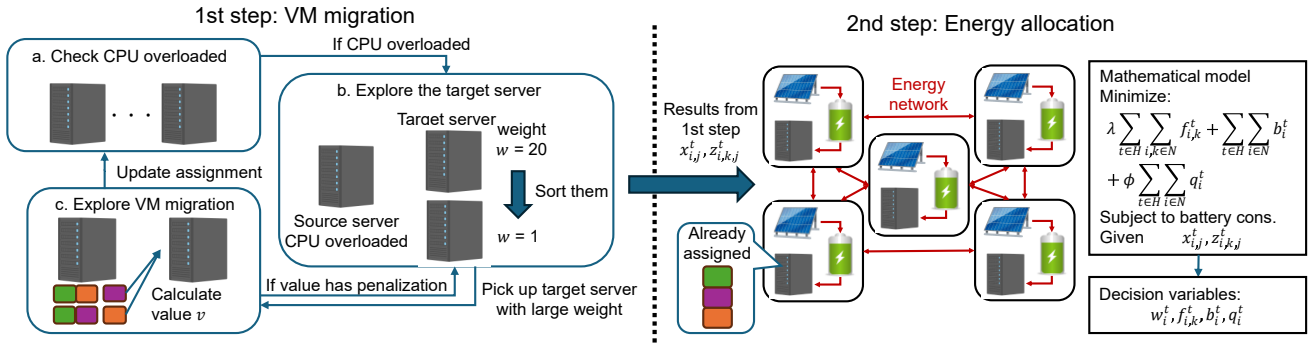


Fig. 3: Overview of two-step optimization.

migration for the current step, the algorithm updates the VM assignment and migration variables (lines 18-23), preparing the system for the next iteration. This two-step approach ensures efficient resource allocation by combining effective target server selection with minimal-cost VM migration.

The computational time complexity of the algorithm is $O(H \cdot N^3 \cdot 2^{V'})$, where H is the planning period, N is the number of nodes, and V' is the number of VMs assigned to a node. In practice, with V' distributed across nodes, this reduces to approximately $O(N^3)$, ensuring scalability.

The algorithm is designed to minimize the number of VM migrations, as frequent migrations increase energy overhead and reduce overall efficiency. Therefore, when no CPU overloading is observed, migrations are generally avoided. The weight factor w favors target servers with greater remaining CPU capacity, larger PV panel area, and larger battery capacity. Prioritizing such servers not only accommodates incoming workloads more reliably but also increases the likelihood that the workload will be powered by locally generated renewable energy, thereby reducing future brown energy purchases. The migration cost v is computed with the aim of selecting VMs whose future CPU utilization fluctuations are small, and doing so in the smallest number necessary to resolve overload situations. This strategy reduces the probability of additional migrations in later time steps, further lowering both migration-related energy losses and operational instability.

B. Second-step Optimization based on LP

Given a VM migration schedule resulting from the prior step, the second step determines energy migration schedule over time based on LP. The corresponding formulation focuses on three main objectives, i.e., reducing energy transfer costs between servers, minimizing the amount of external (brown) energy purchased from the grid, and reducing losses when excess renewable energy is injected back into the grid:

$$\text{Minimize } \lambda \sum_{t\in H}\sum_{i,k\in N} f^t_{i,k} + \sum_{t\in H}\sum_{i\in N} b^t_i + \phi \sum_{t\in H}\sum_{i\in N} q^t_i,$$

subject to constraints (5) - (7),

given $z^t_{i,k,j}, x^t_{i,j}, \quad \forall t,i,k,j,$

with decision variables $w^t_i, f^t_{i,k}, b^t_i, q^t_i, \quad \forall t,i,k,j.$

To achieve the above goals, the model determines the optimal amount of energy to transfer between nodes, purchase from the grid, or inject back into it. The given constraints ensure that resource and energy usage meets the system's requirements, while minimizing the operational costs.

IV. Simulations

A. Setup

Baselines: We compare our two-step approach with the state-of-the-art one [14], as well as three popular greedy algorithms, i.e., FFD [15], BFD [16], and WFD [17]. There are several research on meta-heuristic approaches [18]–[20] and deep learning based methods in [21] and [22] for data center management. However, we excluded these methods due to their high computational costs, which contradict the energy-saving objectives of this study.

Scenarios: We conduct two experimental scenarios to evaluate both energy efficiency and runtime. As for energy efficiency

Algorithm 1: A pseudo code of our greedy approach.

```
Function OptimizeVMmigration():
    for each node i do
        a) Check CPU overloading
        if C_s > R_s then
            Initialize v,v_min
            b) Explore the target server
            Sort nodes k by w (descending)
            for each node k in sorted order do
                if k ≠ i and C_d < R_d then
                    c) Explore VM migration
                    for each combination comb of
                     virtual machines do
                        v ← J_m σ_m + Pe
                        if v <v_min then
                            Update optimal migration
                             and target node
                            v_min ← v
                    if v_min < Pe then
                        break
Initialize variables: x^t_{i,j}, z^t_{i,k,j}
Get U_max, S_max
Evenly assign virtual machines to each node i as
 initial assignment
for t from 1 to T − 1 do
    OptimizeVMmigration()
    Update x^t_{i,j}, z^t_{i,k,j}
```

experiment, we model a 5-node heterogeneous mini data center: three small green nodes equipped each, with two PV solar panels and a 3,000 Wh battery, and two large nodes equipped with seven PV solar panels and an 8,000 Wh battery each. The total number of VMs in the system is 25, five times the number of nodes.

For the runtime evaluation, we consider large-scale systems. Each setup includes twice as many large nodes and three times as many small nodes. For example, a 20-node system contains eight large and 12 small nodes. The number of VMs is consistently set to five times the number of nodes. Each node's facilities, such as solar panels and batteries, are configured according to the specifications in the following paragraphs. All nodes are connected via an energy network directly. For data networking, each node is connected only to the central data server. For the sake of simplicity, VMs and energy migrations consume power, but do not incur latency during transfers in the model.

Parameter settings: We consider the same parameter settings as [14], with key values provided here for clarity. The time step τ is set to five minutes, and the total simulation period comprises $H = 2,016$ time steps, representing a full seven-day period (calculated as 60 minutes per hour $\times$ 24 hours per day $\times$ seven days $\div$ five minutes per time step). The energy transfer parameters are defined as follows: the total energy cost μ is set to 1.17 Wh, with source and target server costs split equally, yielding $\mu_d = \mu/2 = 0.585$ Wh and $\mu_s = \mu/2 = 0.585$ Wh [25]. The penalty for CPU overloading, ν, is set to 1,000, and the energy loss factor ϕ for grid injection is set to 0.5. Node-specific parameters include a standby power ϵ_s of 4.17 Wh (equivalent to 50 W) and a peak power ϵ_p of 37.50 Wh (450 W). The initial SoC for each node's battery is set to $I_i = 0.5 \times U_i$.

Historical solar irradiation data is obtained from the European Photovoltaic Geographical Information System (PVGIS) [26]. We used hourly irradiation data for Montpellier, France (Latitude 43.611, Longitude 3.876) from 2005 to model solar energy generation for each node. This data was pre-processed using piecewise linear interpolation to convert from an hourly basis to the five-minute time interval, τ. VM workloads were simulated using real workload traces from the CoMon project, a monitoring infrastructure for PlanetLab [27]. The PV panels in this model have an area ρ_s of 1.59 m^2 and a conversion efficiency ρ_e of 0.19. The energy generated from PV power, denoted as G_i^t and measured in Watt-hours (Wh), is calculated by the following equation, which considers solar radiation ι^t, the number of PV panels ρ_n, the area per PV panel ρ_s, and the conversion efficiency ρ_e:

$$G_i^t = \iota^t \times (\rho_n \times \rho_s \times \rho_e) \times \frac{\tau}{3,600} \tag{8}$$

where τ represents the time step resolution.

Implementation: The aforementioned formulations and algorithms are implemented in Julia (version 1.6.5) using JuMP [28] and executed on an Intel(R) Xeon(R) CPU E7-8890 v4 2.20 GHz, with 96 cores. Gurobi 11.0.2 is used as the LP and MILP solver. Figure 4 illustrates and compares the optimization processes of the standard MILP, the rolling-horizon MILP, and the proposed two-step Greedy–LP approach over the entire planning horizon. In the standard MILP method, VM and energy optimization are jointly solved for the whole horizon, achieving global optimality but with extremely high computational cost. In the rolling-horizon MILP method, the planning horizon is divided into overlapping 24-step windows to reduce computational complexity. Each window is allotted a time limit of 600 seconds (10 minutes). If the optimal solution is not reached within this period, the current solution is passed as the initial condition to the subsequent window. This window-based process reduces runtime but produces heuristic results, as decisions in early windows constrain subsequent ones. In contrast, the proposed two-step approach decouples VM scheduling and energy allocation without applying a rolling horizon. VM migrations are first determined for the entire planning horizon using a greedy algorithm, and the resulting VM placements and migration plans are then used as fixed inputs for the subsequent LP-based energy allocation. As with the MILP methods, the proposed approach produces heuristic—but far more scalable—solutions.

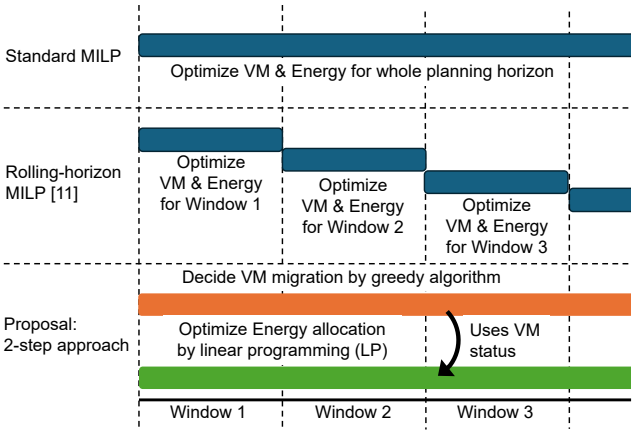


Fig. 4: Comparison of optimization flows: standard MILP, rolling-horizon MILP, and the proposed two-step Greedy–LP approach.

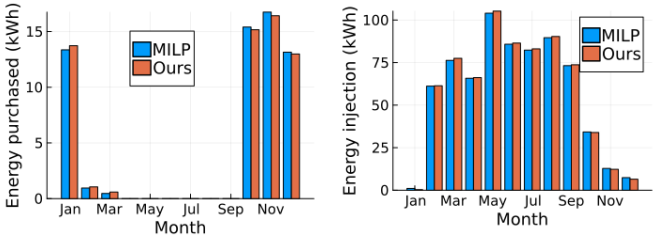
TABLE II: Average of annual results for each method.

	MILP	Two-step approaches			
		FFD	BFD	WFD	Ours
Computational time [sec]	21,240 ≒ 5.9h	1.027	1.009	0.948	**0.913**
Number of VM migration	429.2	50,400	50,400	50,400	**259.0**
Energy transferred [kWh]	24.1	159.1	159.0	62.5	**23.7**
Purchased energy [kWh]	5.01	32.38	32.38	25.01	**4.99**
Energy injection [kWh]	57.8	12.2	12.2	17.8	**58.1**

B. Results

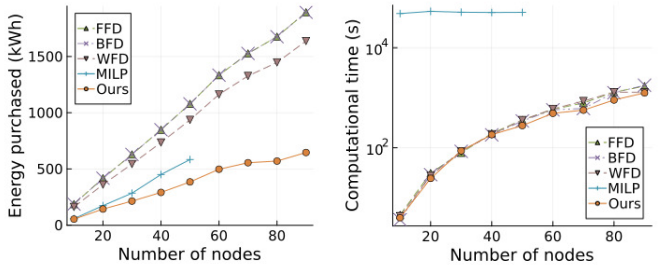
Energy efficiency comparison: Table II compares results from the state-of-the-art MILP method [14] and the two-step approaches including our proposed solution, referred to as "Ours". In the two-step approaches, the first step uses FFD, BFD, and WFD for VM migrations, and then the described LP-based method in the the second step, as presented in Section III-B. Our method reduces the number of VM migrations by 39.7% and energy transfers by 1.7% compared with MILP method. It has resulted in 0.4% less purchased brown energy and 0.5% more energy injection due to the reduction in overall system losses. The MILP baseline in [14] is solved using a rolling-horizon approach, where the planning horizon is divided into shorter windows and optimized sequentially. While this reduces computation time, it can lead to suboptimal global results because early-window decisions constrain later ones. In contrast, our method determines VM migrations and energy allocation over the entire horizon in a single run, avoiding inter-window constraints. This difference in optimization scope explains why, despite the MILP model being optimal within each window, our approach achieves slightly lower purchased energy in some cases.

Figures 5a and 5b show monthly results for purchased and injected energy for the MILP model and our method. In each month, our algorithm has differences from -0.33 kWh to 0.37 kWh in purchased energy and from -0.92 kWh to 1.23 kWh in energy injection compared to MILP. Although there are some seasonal variations, the results throughout the year is almost the same as the MILP model solution.



(a) Purchased energy (b) Injected energy

Fig. 5: Monthly optimization results of MILP and our method for a 5-node mini data center.

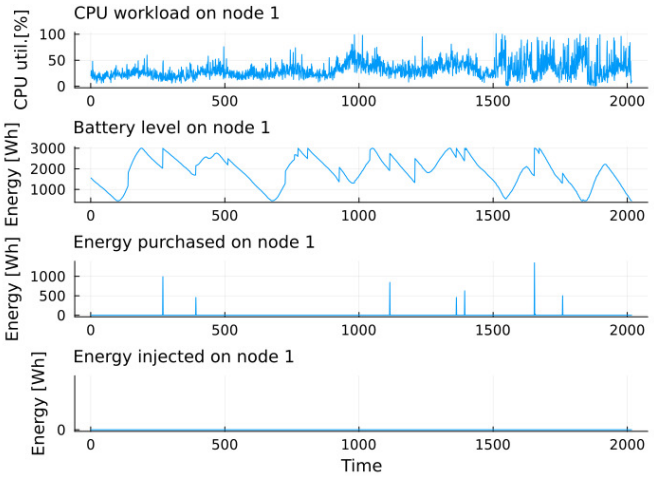


(a) Purchased energy (b) Computational time

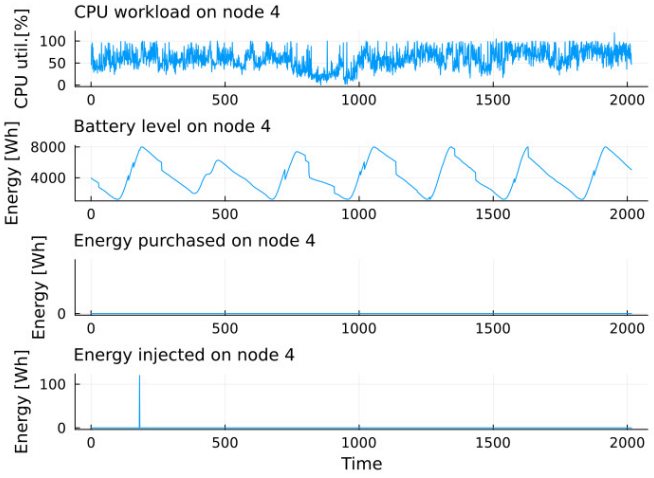
Fig. 6: Comparison of methods for large-scale systems for January.

Runtime comparison: From Table II, our solution is up to 12.1% faster than other traditional VM assignment algorithms. Notably, it is 99.6% faster than the MILP method. Figures 6a and 6b show the purchased energy and computational time according to the number of green nodes. Compared to the traditional VM allocation methods, our solution saves up to 34% computation time and 72% purchased brown energy. The MILP model struggles with computation time as the number of nodes increases. For 10 nodes, it fails to find a solution within the time limit, extending up to 50,000 seconds (i.e., 15.8 hrs). As nodes increase, computation time remains high, but solution quality declines. At 50 nodes, purchased energy is 51.7% higher than our algorithm, and with 60 nodes, no solution is found within the time limit. On the other hand, our approach is able to find an optimal solution within 25 minutes even when the number of nodes is 90, while achieving a small purchased energy.

Mini data centers, which are the focus of this study, are generally expected to have a relatively small number of compute nodes. Therefore, 90 nodes can be considered a pessimistic upper bound, at most mini data centers are likely smaller in size. This highlights the robustness and practicality of the proposed method, even under larger-than-usual scenarios.



(a) Node 1: small facilities.



(b) Node 4: large facilities.

Fig. 7: Workload and energy profiles on small and large nodes for January.

C. Discussion and possible limitation of our approach

VM allocation behavior and limitations: Figure 7a, 7b show the obtained results with our solution. Our method determines the allocation of VMs to nodes with larger facilities without exceeding 100% server utilization at the nodes. It has been designed to keep the smaller nodes' SoC high and their purchased energy low. As a result, Node 1 sustains a high SoC most of the planning horizon, whereas Node 4, with larger capacity, has energy injection.

Figure 8 illustrates the assignment of each VM in the 5-node system over time. Gray color represents nodes 1, 2, and 3, which are smaller facilities, while blue color denotes the larger facilities, nodes 4 and 5. Approximately half of the VMs are allocated to nodes 4 and 5 throughout the planning period, reducing unnecessary energy transfers. For instance, VM_5 has a CPU utilization rate that fluctuates between 0% and 5%, as depicted in the Figure 2b. Since its initial allocation to node 1, it has remained on the same node, indicating that the allocation strategy minimized VM migration while avoiding CPU overload. Conversely, VM_2 and VM_3 exhibit more frequent node changes midway through the process. This behavior stems from sudden spikes in their CPU utilization, jumping from a range of 20% to 30% to a sudden peak at 80%. Such fluctuations could not be anticipated based on standard deviation alone, making them challenging to manage. Thus, there is a limitation in handling such dynamic behavior, especially in scenarios involving unpredictable workload fluctuations or environmental changes. Addressing these challenges requires adaptive strategies such as anomaly detection, which play a critical role in identifying and responding to irregularities in real time. These strategies are essential for improving the stability and operational efficiency of volatile data center environments.

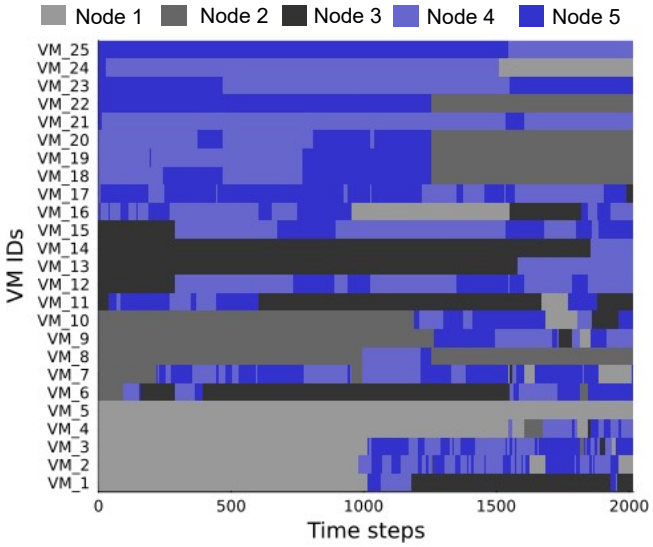


Fig. 8: VM assignment results for five nodes, with our method.

Parameter sensitivity in energy and VM scheduling: To evaluate the robustness of the proposed method under varying system conditions, we conducted a sensitivity analysis on key parameters, including photovoltaic (PV) generation capacity, battery capacity, and the energy cost per unit of VM migration (μ).

Figure 9 shows the relative changes in purchased energy, VM migrations, and computation time when each parameter was varied from its baseline. The number of VM migrations, determined by the first greedy allocation step, and the computation time, governed by the fixed greedy–LP workflow, remained unchanged across all scenarios. In contrast, purchased energy was sensitive to reduced PV or battery capacity, which increased grid energy use. Likewise, a higher VM migration energy cost (μ) raised purchased energy due to the LP's trade-off between migration loss and grid procurement. Overall, while VM allocation and runtime are robust to parameter variations, the LP-based scheduling is affected by renewable generation and storage capacity, underscoring the need for sufficient PV and battery resources.

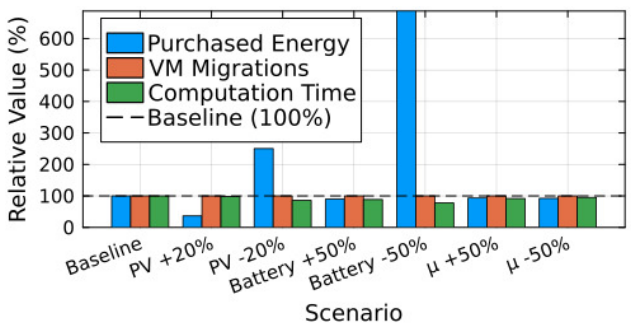


Fig. 9: Sensitivity analysis of purchased energy, VM migrations, and computation time for variations in PV capacity, battery capacity, and VM migration energy cost (μ) relative to the baseline.

V. Conclusions

This paper presented a two-step approach to determine suitable data and energy migrations for scalable and carbon-neutral mini data centers. The proposed solution tackles the computational bottleneck associated with solving the VM migration optimization problem by introducing a novel greedy algorithm. Experimental results demonstrate that our approach consistently outperforms existing VM allocation methods, through a 99.6% reduction in computational time to solve the VM migration problem, while preserving a high energy efficiency compared with these methods. Notably, our solution allows for handling design scenarios 18 times larger than a state-of-the-art MILP-based method.

In the future, we plan to reduce static power consumption by shutting down underutilized nodes and migrating VMs efficiently, while exploring adaptive strategies for abrupt workload changes. Extending our simplified VM model beyond CPU utilization is also valuable future work.

References

[1] N. Jones, "How to stop data centres from gobbling up the world's electricity," *Nature*, vol. 561, no. 7722, pp. 163–166, 2018.

[2] E. Masanet, A. Shehabi, N. Lei, S. Smith, and J. Koomey, "Recalibrating global data center energy-use estimates," *Science*, vol. 367, no. 6481, pp. 984–986, 2020.

[3] E. Sage, "Renewable energy credits," 2020. Accessed: Dec. 23, 2020. [Online]. Available: https://www.energysage.com/otherclean-options/renewable-energy-credits-recs/

[4] Google, "Google environmental report," 2019. Accessed: Nov. 15, 2024. [Online]. Available: https://www.gstatic.com/gumdrop/sustainability/google-2019-environmental-report.pdf

[5] Facebook, "Facebook sustainability," 2021. Accessed: Nov, 15, 2024. [Online]. Available: https://sustainability.fb.com

[6] Amazon, "Sustainable operations: Renewable energy," 2021. Accessed: Nov. 16, 2024. [Online]. Available: https://sustainability.aboutamazon.co.uk/environment/sustainable-operations

[7] F. Kong and X. Liu, "A survey on green-energy-aware power management for datacenters," *ACM Computing Surveys*, vol. 47, no. 2, pp. 1–38, 2015.

[8] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *IEEE INFOCOM*, pp. 1-9, 2010.

[9] L. Wang et al., "GreenDCN: A general framework for achieving energy efficiency in data center networks," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 1, pp. 4–15, 2014.

[10] S. K. Mishra, D. Puthal, B. Sahoo, P. P. Jayaraman, S. Jun, A. Y. Zomaya, and R. Ranjan, "Energy-efficient VM-placement in cloud data center," *Sustainable Computing: Informatics and Systems*, vol. 20, pp. 48–55, 2018.

[11] D.-M. Zhao, J.-T. Zhou, and K. Li, "An energy-aware algorithm for virtual machine placement in cloud computing," *IEEE Access*, vol. 7, pp. 55659–55668, 2019.

[12] M. Xu, A. N. Toosi, and R. Buyya, "A self-adaptive approach for managing applications and harnessing renewable energy for sustainable cloud computing," *IEEE Transaction on Sustainable Computing*, vol. 6, no. 4, pp. 544–558, 2021.

[13] W. He, Q. Xu, S. Liu, T. Wang, F. Wang, X. Wu, Y. Wang, and H. Li, "Analysis on data center power supply system based on multiple renewable power configurations and multi-objective optimization," *Renewable Energy*, vol. 222, p. 119865, 2024.

[14] Marcos De Melo da Silva, Abdoulaye Gamatié, Gilles Sassatelli, Michael Poss, and Michel Robert, "Optimization of data and energy migrations in mini data centers for carbon-neutral computing," *IEEE Transactions on Sustainable Computing*, vol. 8, no. 1, pp. 68-81, 2022.

[15] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing SLA violations," in *IFIP/IEEE International Symposium on Integrated Network Management*, pp. 119–128, 2007.

[16] A. Varasteh and M. Goudarzi, "Server consolidation techniques in virtualized data centers: A survey," *IEEE Systems Journal*, vol. 11, no. 2, pp. 772–783, 2017.

[17] S. Dhahbi, M. Berrima, and F. A. M. Al-Yarimi, "Load balancing in cloud computing using worst-fit bin-stretching," *Cluster Computing*, vol. 24, no. 4, pp. 2867–2881, 2021.

[18] T. P. Shabeera, S. D. Madhu Kumar, S. M. Salam, and K. Murali Krishnan, "Optimizing VM allocation and data placement for data-intensive applications in cloud using ACO metaheuristic algorithm," *Engineering Science and Technology International Journal*, vol. 20, no. 2, pp. 616–628, 2017.

[19] S. Supreeth, K. Patil, S. D. Patil, and S. Rohith, "Comparative approach for VM scheduling using modified particle swarm optimization and genetic algorithm in cloud computing," in *IEEE International Conference on Data Science and Information System (ICDSIS)*, vol. 10, pp. 1–6, 2022.

[20] A. K. Singh, S. R. Swain , D. Saxena, C.-N. Lee, "A bio-inspired virtual machine placement toward sustainable cloud resource management," *IEEE Systems Journal*, vol. 17, no. 3, pp. 3894-3905, 2023.

[21] D. Saxena and A. K. Singh, "A proactive autoscaling and energy-efficient VM allocation framework using online multi-resource neural network for cloud data center," *Neurocomputing*, vol. 428, pp. 248-264, 2021.

[22] K. Sathupadi, "Deep Learning for Cloud Cluster Management: Classifying and Optimizing Cloud Clusters to Improve Data Center Scalability and Efficiency," *Data Analytics and Cloud Computing*, vol. 6, no. 2, pp. 33-49, 2021.

[23] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2011.

[24] X. Liu, J. Wu, G. Sha, and S. Liu, "Virtual machine consolidation with minimization of migration thrashing for cloud data centers," *Mathematical Problems in Engineering*, vol. 2020, pp. 1–13, 2020.

[25] H. Liu, C.-Z. Xu, H. Jin, J. Gong, and X. Liao, "Performance and energy modeling for live migration of virtual machines," *Proceedings of the 20th International Symposium on High Performance Distributed Computing*, pp. 171–182, 2011.

[26] The European Commission's science and knowledge service, "Photovoltaic geographical information system (PVGIS)," 2021. Accessed: Nov. 18, 2024. [Online]. Available: https://ec.europa.eu/jrc/en/pvgis

[27] K. Park and V. S. Pai, "CoMon: A mostly-scalable monitoring system for PlanetLab," *ACM SIGOPS operating systems review,* vol. 40, no. 1, pp. 65–74, 2006.

[28] I. Dunning, J. Huchette, and M. Lubin, "JuMP: A modeling language for mathematical optimization," *SIAM Review*, vol. 59, no. 2, pp. 295–320, 2017.