Computing and Artificial Intelligence 2026, 4(2), 4439.
doi: https://doi.org/10.59400/cai4439

Article

Static vs. dynamic cloud scheduling: A benchmark driven performance and energy study

Saba Naz, Altaf Hussain*

Department of Computer Science, Institute of Space Technology, KICSIT Campus, Islamabad 44000, Pakistan
*** Corresponding author:** Altaf Hussain, altaf.hussain@ist.edu.pk

CITATION

Naz S, Hussain, A. Static vs. dynamic cloud scheduling: A benchmark driven performance and energy study. Computing and Artificial Intelligence. 2026; 4(2): 4439. https://doi.org/10.59400/cai4439

ARTICLE INFO

Received: 13 February 2026
Revised: 20 April 2026
Accepted: 22 April 2026
Available online: 7 May 2026

Abstract: Major multinational corporations, including Amazon, Microsoft, IBM, and Google, continue to advance network and computational infrastructures through the deployment of highly efficient data centers worldwide to strengthen their cloud computing services. Despite these technological advancements, cloud computing environments still face several critical challenges, such as optimal resource utilization, cost efficiency, security, fault tolerance, scalability, and energy consumption. For organizations operating private clouds or delivering cloud-based services, the primary objectives include maximizing resource utilization, minimizing response and waiting times, increasing throughput and profitability, and enhancing the overall user experience, with energy efficiency being a particularly important concern. This study presents an empirical evaluation of various static and dynamic cloud job scheduling algorithms using the CloudSimPlus simulation framework. The algorithms are assessed using two well-established scientific benchmark datasets: HCSP and GoCJ instances. Experimental results reveal that the Resource-Aware Load Balancing Algorithm (RALBA) emerges as the most energy-efficient static scheduling approach, achieving reduced makespan, improved resource utilization, and lower energy consumption. In contrast, the Max–Min algorithm exhibits the highest makespan, the lowest throughput, inefficient resource utilization, and the greatest energy consumption. Among the dynamic scheduling techniques, DE-RALBA demonstrates superior resource utilization and competitive makespan performance while maintaining relatively efficient energy usage. However, the results also indicate that dynamic scheduling approaches may incur higher energy consumption depending on workload characteristics and rescheduling overhead. Unlike prior studies that evaluate scheduling strategies in isolation, this work provides a unified, benchmark-driven, and energy-aware comparison of static and dynamic scheduling paradigms, enabling deeper insights into performance–energy trade-offs.

Keywords: cloud computing; job scheduling; energy efficiency; resource utilization; CloudSimPlus simulation

1. Introduction

Cloud computing has fundamentally transformed the way organizations and individuals' access and utilize computational resources. By enabling on-demand access to shared pools of computing power, storage, and networking infrastructure, cloud platforms deliver scalable, flexible, and cost-effective solutions that support a wide range of applications and services [1,2]. Despite these advantages, the rapid expansion and adoption of cloud computing infrastructures introduce significant challenges,

particularly in the efficient management of resources and energy consumption.

Resource utilization in cloud environments refers to the effective allocation and use of physical and virtual resources—such as processing power, memory, storage, and network bandwidth—across distributed systems. Inefficient utilization can lead to resource underuse or overload, resulting in increased operational costs, degraded performance, and reduced service quality. To address these challenges, cloud service providers employ techniques such as virtualization, containerization, and adaptive resource provisioning, which enable dynamic adjustment to fluctuating workloads and improve overall system efficiency [3].

Energy consumption has emerged as another critical concern in modern cloud computing. Large-scale data centers require substantial electrical power to operate and cool thousands of high-performance servers, leading to rising operational costs and significant environmental impact [4]. As cloud services continue to proliferate globally, minimizing the energy footprint of data centers has become a priority from both economic and sustainability perspectives. Consequently, research efforts increasingly focus on strategies that jointly address performance optimization and energy efficiency, including workload consolidation, energy-aware scheduling, and the integration of renewable energy sources. Achieving an effective balance between these factors is essential to ensuring the long-term scalability, reliability, and sustainability of cloud services.

Job scheduling plays a fundamental role in cloud computing, determining how tasks are assigned to available resources to achieve efficient execution. In large-scale and heterogeneous cloud environments, scheduling decisions directly influence system performance, resource utilization, and energy consumption. Suboptimal scheduling may result in server overloading or underutilization, increased processing delays, and excessive power usage. These challenges are exacerbated by the dynamic and unpredictable nature of cloud workloads. Energy-aware scheduling techniques attempt to mitigate these issues by consolidating tasks onto fewer active servers, enabling idle resources to transition into low-power or shutdown states. Such approaches significantly reduce energy consumption while maintaining acceptable performance levels, making them essential for the development of sustainable cloud infrastructures [5,6].

Within a data center, numerous servers operate concurrently to process user requests, which may be executed sequentially or in parallel. When a server becomes overloaded, tasks must be redirected to alternative capable servers to maintain service quality [7]. Scheduling algorithms in distributed systems aim to divide workloads and allocate subtasks across processors in a way that maximizes resource utilization and minimizes execution time. However, task migration and scheduling across geographically distributed data centers can increase energy consumption and carbon dioxide (CO_2) emissions [4]. The selection of an effective scheduling algorithm is therefore crucial yet inherently complex due to workload diversity, resource heterogeneity, and competing optimization objectives [8].

Although estimating overall data center energy consumption is relatively straightforward, fine-grained monitoring of energy usage at the level of individual

nodes is essential for informed scheduling decisions and load balancing [9]. Modern cloud data centers face substantial energy demands not only from computing operations but also from supporting infrastructure, including cooling systems, storage units, and networking equipment [10]. Given that cloud platforms may consist of thousands of high-performance computing hosts, inefficient scheduling can substantially increase total energy consumption [11]. Consequently, selecting an appropriate scheduling algorithm requires careful consideration of workload characteristics, system heterogeneity, and trade-offs among makespan, throughput, resource utilization, and energy efficiency [8].

This study investigates cloud job scheduling from the perspectives of makespan, resource utilization, and energy consumption to evaluate the effectiveness of different scheduling strategies in executing workloads efficiently.

Contributions and novelty

Although this study does not propose a new scheduling algorithm, its novelty lies in delivering a comprehensive, standardized, and energy-aware empirical evaluation framework for cloud scheduling research. The key contributions are as follows:

1. **Unified benchmark-driven evaluation framework:** This work presents a systematic and reproducible evaluation framework for both static and dynamic cloud scheduling algorithms using CloudSimPlus. Unlike prior studies that examine these paradigms separately or under heterogeneous conditions, this study ensures fair cross-comparison under identical simulation settings.
2. **Use of realistic scientific benchmark workloads:** The study employs HCSP and GoCJ benchmark datasets, enabling realistic workload modeling. This distinguishes the work from many existing studies that rely on synthetic or simplified task distributions, thereby improving the validity and generalizability of the results.
3. **Holistic multi-metric performance assessment:** A comprehensive evaluation is conducted across makespan, throughput, resource utilization, and energy consumption, providing a multi-dimensional understanding of scheduling behavior that goes beyond single-metric optimization. A key novelty of this work is the explicit characterization of trade-offs between scheduling efficiency and energy consumption.
4. **Quantitative analysis of energy–performance trade-offs:** Demonstrating how dynamic scheduling overhead affects energy usage. This provides new empirical evidence on the cost of adaptability in cloud environments.
5. **Cross-paradigm comparative insights:** The study provides new insights into the relative strengths and limitations of static vs. dynamic scheduling approaches, revealing conditions under which static methods (e.g., RALBA) outperform dynamic ones in energy efficiency despite lower adaptability.
6. **Practical guidelines for cloud system design:** Based on experimental findings, the work offers actionable recommendations for selecting scheduling strategies depending on application requirements such as performance optimization or energy efficiency, making the study valuable for both researchers and

practitioners.

The remainder of this paper is organized as follows: Section 2 describes the scheduling algorithms under consideration, the energy consumption model, the experimental setup, and the benchmark datasets used. Section 3 presents the experimental results, followed by an in-depth discussion in Section 4. Finally, Sections 5 and 6 present limitations and conclude the paper.

2. Materials and methods

This section presents twelve job scheduling techniques, comprising six static and six dynamic approaches, alongside the adopted energy consumption model, the scientific benchmark datasets, namely Google Cloud Jobs (GoCJ) [12] and Heterogeneous Computing Scheduling Problems (HCSP) [13], as well as the computational setup and simulation environment employed in this study.

2.1. Job scheduling in cloud computing

To understand how these approaches work and how they achieve key scheduling goals, the working semantics of scheduling techniques are elaborated.

Task scheduling has long been recognized as a critical research area in distributed and cloud computing due to its direct impact on performance, resource utilization, and energy efficiency. Early scheduling strategies primarily focused on maximizing processor utilization, often without considering system heterogeneity, workload diversity, or energy constraints. Opportunistic Load Balancing (OLB) is among the earliest and simplest scheduling algorithms proposed for distributed environments [4,14–16]. OLB assigns each incoming task to the next available resource with the primary objective of preventing processor idleness. While this approach achieves high resource occupancy, it disregards task characteristics, execution time estimates, and resource heterogeneity. As a result, OLB frequently leads to load imbalance, increased makespan, and inefficient energy usage, particularly in modern heterogeneous and dynamic cloud infrastructures. Nevertheless, OLB remains a commonly used baseline algorithm for comparative evaluation due to its simplicity and low scheduling overhead. It performs relatively well in environments with uniform task sizes or where rapid response is prioritized over execution efficiency.

Heuristic-based static scheduling algorithms such as Min-Min, Max-Min, and Minimum Completion Time (MCT) have been extensively studied for their balance between computational simplicity and scheduling effectiveness in heterogeneous environments. The Min-Min algorithm [17–22] computes the minimum expected completion time for each task across available resources and assigns the task with the smallest such value first. By prioritizing shorter tasks, Min-Min can significantly reduce average completion time and improve system throughput. However, this bias often results in starvation or prolonged delays for larger tasks, especially under imbalanced workloads.

In contrast, the Max-Min algorithm [23–25] assigns priority to tasks with the largest minimum completion time, thereby executing longer tasks earlier. This strategy helps prevent long tasks from becoming bottlenecks and is particularly effective in

environments with high task heterogeneity. However, Max-Min may negatively impact the response time of smaller tasks, increasing the overall average completion time under certain workload distributions.

The MCT algorithm [4,26] adopts a greedy approach by assigning each task to the resource on which it completes fastest, independent of system-wide load conditions. Although MCT is computationally lightweight and easy to implement, its lack of load awareness often results in uneven resource utilization and suboptimal performance in dynamic cloud environments. Despite these drawbacks, MCT serves as a foundational benchmark for evaluating more sophisticated scheduling strategies.

To address fairness and performance degradation caused by suboptimal task placement, the Sufferage scheduling algorithm was introduced [4,15,23,26]. Sufferage prioritizes tasks based on the difference between their best and second-best expected completion times, thereby protecting tasks that would suffer the most if not assigned to their optimal resource. This approach improves makespan and load balance in heterogeneous systems but introduces higher computational overhead due to repeated evaluations and dependency on accurate execution time estimates.

Resource-aware scheduling techniques have gained prominence with the increasing heterogeneity of cloud infrastructures. The Resource-Aware Load Balancing Algorithm (RALBA) [27–29] enhances traditional heuristic scheduling by incorporating both resource capabilities and current load conditions into task assignment decisions. RALBA employs a ranking mechanism to match tasks with appropriate resources based on processing power, memory availability, and utilization status. This enables improved makespan reduction, better load distribution, and enhanced system responsiveness, albeit at the cost of modest scheduling overhead.

Dynamic scheduling approaches further extend resource awareness by continuously adapting to real-time system conditions. Dy-Max-Min [28, 30] dynamically assigns the most computationally intensive tasks to the virtual machines (VMs) with the earliest completion times, updating scheduling decisions as execution progresses. This adaptability improves load balancing and resource utilization under changing workloads.

Simulation-based dynamic algorithms such as Proactive Simulation-Based Scheduling and Load Balancing (PSSLB) [4, 31] and Proactive Simulation-Based Scheduling and Enhanced Load Balancing (PSSELB) [31] employ predictive modelling to estimate task completion times across VMs. PSSELB enhances earlier approaches by incorporating additional system-level parameters, including VM load history and real-time availability, leading to reduced makespan, improved throughput, and balanced resource utilization in large-scale cloud environments.

Dynamic Resource-Aware Load Balanced Scheduling Approaches, including DRALBA [32] and Dynamic Enhanced Resource-Aware Load Balancing Algorithm (DE-RALBA) [33,34], represent advanced scheduling solutions that combine real-time monitoring, dynamic ranking, and proactive workload rebalancing. DE-RALBA extends RALBA by continuously updating VM rankings based on execution state, queue lengths, and resource usage. This enables precise task placement, improved makespan, enhanced energy efficiency, and better service quality in multi-tenant and

highly dynamic cloud environments. However, the continuous monitoring and decision updates introduce moderate computational overhead, which must be carefully managed to preserve scalability.

Overall, prior research demonstrates that while static heuristic algorithms offer low overhead and simplicity, they struggle to cope with dynamic workloads and energy constraints. Dynamic and resource-aware approaches provide superior adaptability and performance but require careful trade-offs between scheduling overhead, scalability, and energy efficiency —motivating the comprehensive evaluation performed in this study. **Table 1** provides detailed implementation-level descriptions of all scheduling algorithms, including their internal decision logic, parameterization, and computational complexity. This expanded description ensures methodological transparency and highlights the operational differences between static and dynamic approaches, particularly in terms of resource awareness, task prioritization, and runtime adaptability.

Table 1. Literature review summary.

Static techniques/algorithms			
Algorithm	**Computation overhead**	**Key idea**	**Implementation note (CloudSimPlus)**
OLB	Very Low O(n)	Assign tasks to the next available VM immediately to maximize utilization but not optimize execution time. Key parameters are none	Immediate dispatch policy
Min-Min	Moderate $O(n^2 \cdot m)$	Computes the completion time of each task on all VMs; selects the task with the minimum completion time for execution first. Key parameters are Task size and VM MIPS	Builds a completion-time matrix
Max-Min	Moderate $O(n^2 \cdot m)$	Selects tasks with maximum completion time and assigns them to the VM that minimizes execution time. Key parameters are Task size and VM MIPS	Builds a completion-time matrix and balances long job execution
MCT	Low–Moderate O(n·m)	Assigns each task to the VM with the minimum expected completion time, independently of other tasks. Key parameters are Task size and VM MIPS	Greedy task-wise assignment
Sufferage	Moderate $O(n^2 \cdot m)$	For each task, computes the difference between its best and second-best VM completion times as the "sufferage value". Tasks with the highest sufferage are scheduled first to avoid poor allocation and performance degradation. Key parameters are Task size, VM MIPS	Requires tracking the first and second best VMs per task
TASA	Moderate $O(n^2)$	Uses adaptive heuristics combining execution time estimation and resource capabilities to balance the workload and improve throughput. Key parameters are Task size, VM capacity	Hybrid heuristic
RALBA	Low to Moderate $O(m^2 \cdot n)$	Assigns tasks based on both VM computational capacity (MIPS) and current load conditions. It computes a load index or utilization metric for each VM and distributes tasks to balance load while minimizing completion time and avoiding overloading high-capacity nodes. Key parameters are VM MIPS, load index, and utilization threshold.	Custom scheduler: evaluates VM load + capacity before assignment; no runtime rescheduling

Table 1. *Cont.*

Dynamic techniques/algorithms			
Algorithm	**Computation overhead**	**Key idea**	**Implementation note (CloudSimPlus)**
Dy-Max-Min	O(k·n^2) k: rescheduling iterations	Dynamic version of Max-Min; adapts to runtime system changes. Key parameters are task size and VM state	Iterative rescheduling
Dy-Min-Min	O(k·n^2)	Recalculates completion times dynamically after each scheduling event. Key parameters are Task size, VM status	Periodic re-computation
PSSLB	Moderate O(n^2)	Simulation-based predictive scheduling. Key parameters are Task size, VM MIPS, and ECT matrix	Requires computation of the ECT matrix similar to Max-Min; prioritizes large jobs
PSSELB	High O(n^2) with additional comparisons	Enhanced simulation with system metrics. Key parameters are Task size, VM MIPS, and EFT pivot value	Extends PSSLB with pivot-based selection; requires tracking completion-time thresholds
DRALBA	O(k·n^2)	dynamic extension of RALBA with aggressive runtime task reassignment based on real-time system metrics to maximize adaptability. Key parameter is VM load, rescheduling rate	Frequent task migration
DE-RALBA	Moderate O(n^2)	Extends RALBA with controlled runtime rescheduling based on updated VM load. The key parameter is VM load thresholds	Hybrid static-dynamic behavior

2.2. Energy consumption model

Energy efficiency has become a critical challenge for modern data centers due to the rapidly increasing demand for cloud services and the associated operational and environmental costs [9,35–37]. Among the various infrastructure components, server machines account for the largest proportion of total energy consumption, with power usage closely tied to utilization levels and workload characteristics. Consequently, accurate modeling of server energy consumption is essential for evaluating the effectiveness of scheduling and resource management strategies in cloud environments.

Data center energy consumption is commonly characterized by two major components: static energy consumption, which is incurred even when resources are idle, and dynamic energy consumption, which results from active workload execution [38]. Empirical studies have shown that servers consume a substantial fraction—often up to 70%—of their peak power even in idle states, primarily due to baseline operational requirements of CPUs, memory, storage, and networking subsystems [37]. This behavior highlights the importance of incorporating idle power costs into energy-aware scheduling models.

In this study, server power consumption is modeled as a linear function of CPU utilization, an approach widely adopted in cloud energy modeling due to its simplicity and empirical validity [39,40].

$$P(U) = P_{Idle} + (P_{max} - P_{Idle}) \times U, \tag{1}$$

where $P_{\max}$ denotes the power consumed at full utilization, P_{Idle} represents idle-state power consumption, and $U \in [0,1]$ is the CPU utilization ratio. In line with prior benchmarks and empirical measurements, the idle power is assumed to be approximately 70% of the maximum power. Consistent with SPECpower benchmark reports, the maximum server power is set to 250 W, representing a realistic average for contemporary server configurations.

The model distinguishes between static and dynamic power consumption components. Static power corresponds to baseline energy usage that remains independent of workload intensity, while dynamic power scales proportionally with CPU and memory utilization associated with virtual machine (VM) execution [38]. Although multiple hardware components contribute to dynamic power consumption, this study focuses primarily on CPU-driven energy consumption, as the CPU remains the dominant energy consumer in cloud servers.

Total energy consumption is computed by integrating power usage over the workload execution interval, defined by the scheduling start and completion times. Energy is measured in watt-hours (Wh), which allows direct comparison across different scheduling strategies and workload configurations. This utilization-based energy model enables an accurate yet tractable estimation of energy consumption as a function of scheduling behavior, making it suitable for large-scale simulations conducted using CloudSimPlus.

The adopted model captures essential energy-performance trade-offs without introducing excessive computational complexity. It provides a consistent basis for quantifying the energy impact of static and dynamic scheduling algorithms evaluated in this study, supporting comparative analysis across workload scenarios and benchmark datasets.

2.3. Computing setup and datasets

The experimental evaluation is conducted using the CloudSimPlus simulation toolkit, where user workloads are modeled as *cloudlets* and their computational requirements are expressed in terms of Million Instructions (MI). All experiments are executed on a system equipped with an Intel Core i5-4200U processor operating at 1.60 GHz (2,301 MHz).

The cloud simulation environment consists of a single data center hosting 30 physical machines (PMs). For the benchmark workloads, 50 virtual machines (VMs) are instantiated for the Google Cloud Jobs (GoCJ) dataset, while 16 VMs are deployed for the Heterogeneous Computing Scheduling Problems (HCSP) instances. The computational capacities of the VMs are specified in Million Instructions Per Second (MIPS). Two well-established scientific benchmark datasets, namely GoCJ and HCSP, are utilized to evaluate the performance of the scheduling algorithms.

2.3.1. GoCJ dataset

The Google Cloud Jobs (GoCJ) dataset [12] reflects realistic workload characteristics derived from Google cluster traces [41–45] and MapReduce execution logs collected from the M45 supercomputing cluster. Owing to its close alignment with production-scale cloud environments, the GoCJ dataset has been widely adopted

in research on cloud scheduling and cloud-based application performance evaluation. The composition and characteristics of cloudlets within the GoCJ dataset are detailed in Hussain and Aleem [12], as illustrated in **Figure 1**. This dataset is widely used in cloud scheduling [32,46–48].

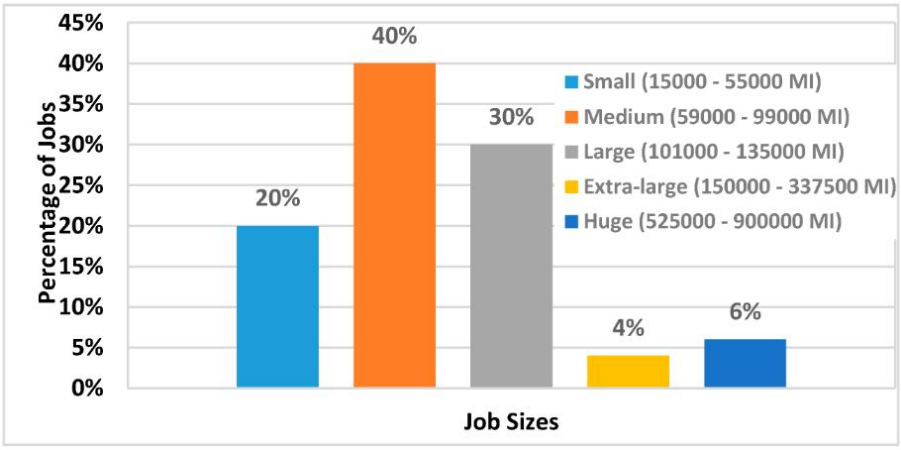

Figure 1. Composition of the GoCJ dataset.

2.3.2. HCSP dataset

The second dataset employed in this study consists of Expected Time to Compute (ETC) models represented as Heterogeneous Computing Scheduling Problem (HCSP) instances [13]. Each HCSP instance is identified using a naming convention of the form *C_T_HM_MH*, where *C* denotes the consistency type—consistent (*c*), semi-consistent (*s*), or inconsistent (*i*). The parameters *T_H* and *M_H* represent task-level and machine-level heterogeneity, respectively, categorized as low (*lo*) or high (*hi*).

The HCSP dataset provides a diverse set of benchmark instances that capture varying degrees of heterogeneity in both job sizes and machine processing capabilities. These characteristics make HCSP particularly suitable for evaluating the robustness and scalability of scheduling algorithms under heterogeneous computing environments. The experimental evaluations are conducted using the same computing configurations adopted in Braun et al. [13], with the corresponding system setups illustrated in **Figures 2** and **3**.

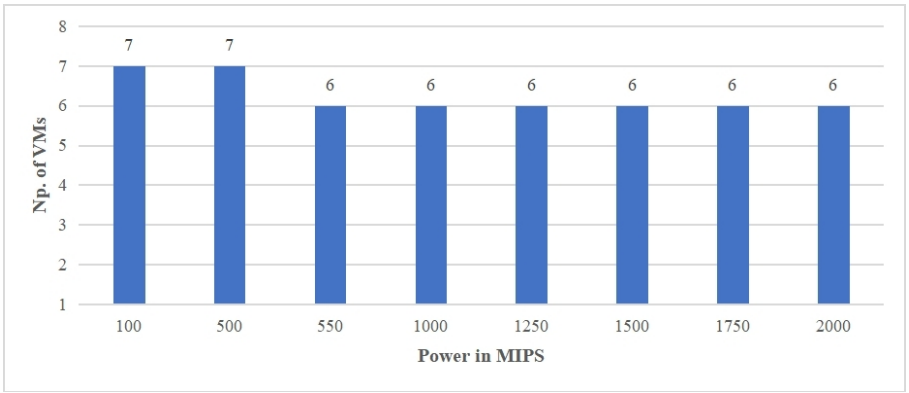

Figure 2. Computing powers of heterogeneous VMs employed for the GoCJ dataset.

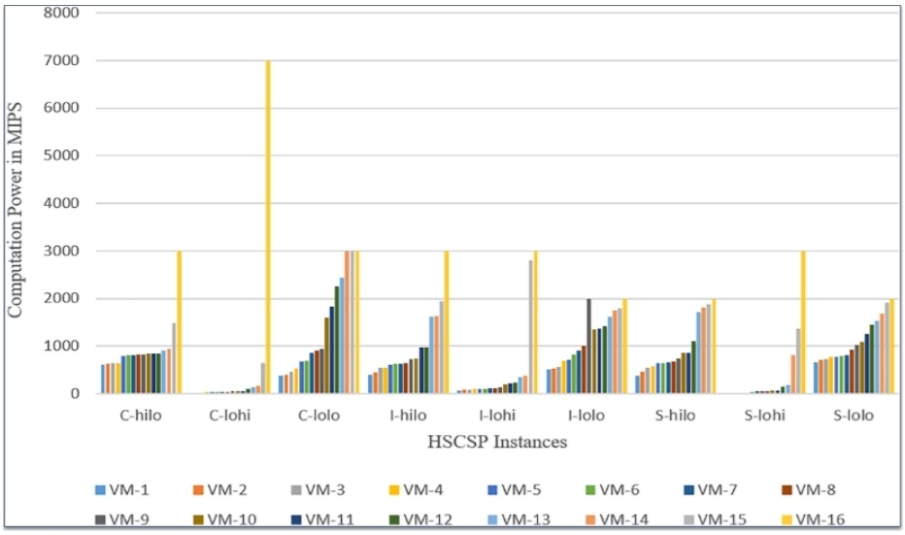

Figure 3. Computing powers of heterogeneous VMs employed for the HCSP dataset.

All benchmark jobs were modeled as individual cloudlets without aggregation to preserve workload granularity. The host-level infrastructure was configured using the default CloudSimPlus settings, ensuring consistency with prior simulation-based studies while providing a standardized and unbiased evaluation environment. The experiments were conducted within a single data center to enable controlled comparison, although the selected benchmark datasets inherently capture workload heterogeneity and scalability characteristics. The simulated data center consists of 30 physical machines (hosts) configured using the default CloudSim Plus settings, where all hosts have identical baseline resource specifications (CPU, memory, storage, and bandwidth). Virtual machines (VMs), however, are heterogeneous in terms of processing capacity (MIPS), as depicted in **Figures 2** and **3** for the GoCJ and HCSP datasets, respectively. The allocation and number of VMs per host are determined dynamically based on available resources, rather than a fixed uniform distribution, to better reflect realistic cloud deployment conditions.

In this study, each job from the GoCJ and HCSP datasets is modeled as an independent cloudlet. While real-world cloud applications often consist of multiple interdependent tasks forming workflows or Directed Acyclic Graphs (DAGs), the selected benchmark datasets do not explicitly provide task dependency information. Therefore, representing each job as a single cloudlet preserves the original dataset characteristics and avoids introducing synthetic assumptions about inter-task dependencies. This modeling choice is consistent with the commonly adopted Bag-of-Tasks (BoT) abstraction used for evaluating baseline scheduling strategies in cloud environments. In addition, Cloudlet arrival times in the simulation are derived from the ordering of jobs in the GoCJ and HCSP benchmark datasets. Specifically, jobs are submitted sequentially according to their sequence in the dataset, and corresponding cloudlets are injected into the simulation environment following this order. This preserves the original workload characteristics without introducing synthetic arrival distributions.

3. Experimentation and results

This section covers the experimentation of static and dynamic scheduling algorithms. The experiments are conducted based on performance parameters such as makespan, resource utilization, and energy consumption.

Each experiment is repeated ten times under identical simulation settings to ensure consistency and eliminate stochastic bias. Due to the deterministic nature of the CloudSimPlus environment and negligible variation across runs, mean values are reported as representative performance metrics, which is consistent with standard practice in simulation-based scheduling studies.

3.1. Makespan results—GoCJ dataset

Figures 4 and **5** present the makespan and average makespan results obtained for the static scheduling algorithms using the GoCJ dataset. In these figures, the x-axis represents the scheduling algorithms, while the y-axis denotes the makespan, measured in seconds, for different groups of cloudlets. Makespan reflects the total time required to complete all assigned tasks; therefore, a lower makespan indicates superior scheduling performance.

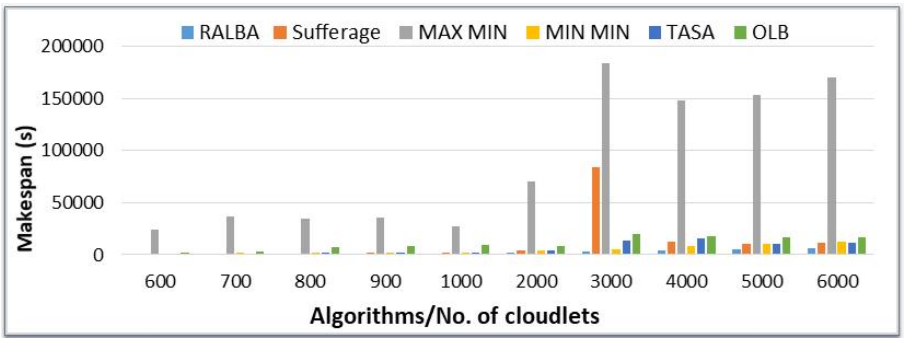

Figure 4. Makespan using GoCJ—Static algorithms.

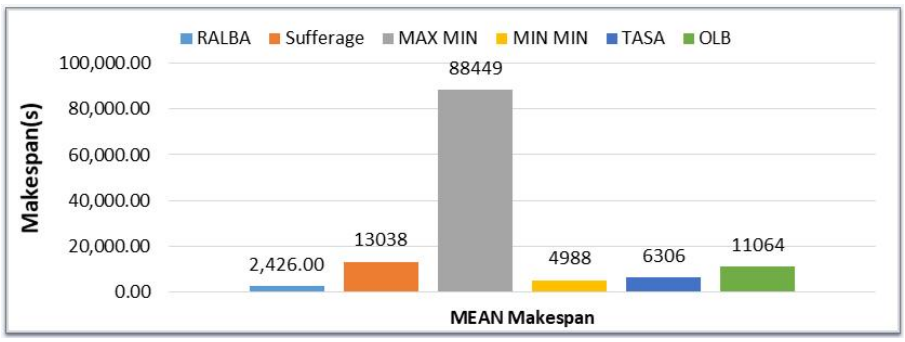

Figure 5. Average Makespan using GOCJ—Static algorithms.

To facilitate clearer interpretation, **Figure 5** illustrates the average makespan values for each static scheduling algorithm. The results reveal that the Max–Min algorithm exhibits the highest makespan across all cloudlet size categories, indicating comparatively poor performance due to longer task completion times. In contrast, RALBA, Min–Min, TASA, and OLB achieve substantially lower makespan values, demonstrating more efficient task execution. Notably, RALBA outperforms all other static algorithms, attaining the lowest average makespan of 2,426 s, thereby ensuring

the fastest overall task completion.

Figures 6 and **7** present the makespan and average makespan results for the dynamic scheduling algorithms. As shown in these figures, the DE-RALBA algorithm consistently achieves the shortest makespan across all cloudlet size categories, indicating superior efficiency in completing tasks within minimal time. DRALBA demonstrates moderate performance, whereas Dy-Max-Min and Dy-Min-Min exhibit the highest makespan values, rendering them the least efficient dynamic approaches.

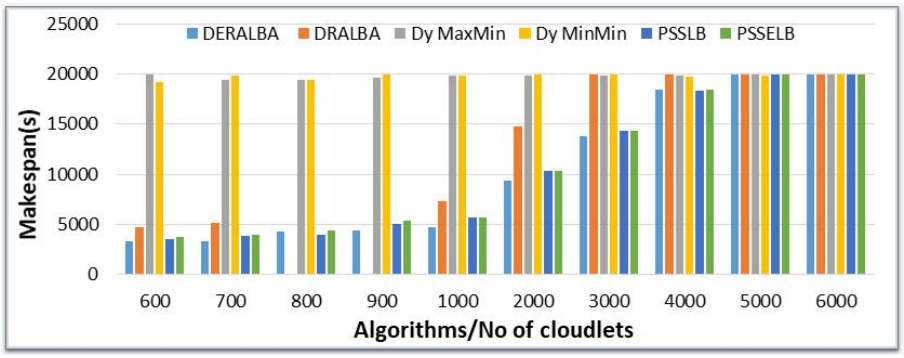

Figure 6. Makespan using GoCJ—Dynamic algorithms.

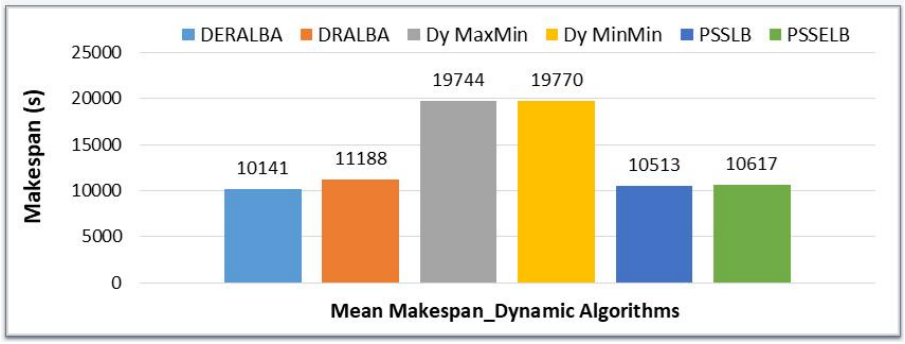

Figure 7. Average Makespan using GOCJ—Dynamic algorithms.

For clearer interpretation, **Figure 7** illustrates the average makespan results for all dynamic scheduling algorithms. DE-RALBA performs best, achieving the lowest average execution time, followed by DRALBA, PSSLB, and PSSELB, respectively. In contrast, Dy-Max-Min and Dy-Min-Min record the highest average makespan, highlighting their relatively poor performance in dynamic scheduling scenarios.

3.2. Average resource utilization ratio (ARUR) results—GoCJ dataset

The average resource utilization ratio (ARUR) values range from 0.0 to 1.0, where higher values indicate more effective utilization of cloud resources. **Figures 8** and **9** present the ARUR and mean ARUR results for the static scheduling algorithms. Among the evaluated approaches, RALBA achieves the highest ARUR, demonstrating superior resource utilization, while TASA also exhibits strong performance. Sufferage and Min–Min show moderate utilization levels, with ARUR values lower than those of RALBA and TASA. In contrast, OLB results in relatively low resource utilization, and Max–Min produces the poorest performance, yielding the lowest ARUR values among the static algorithms.

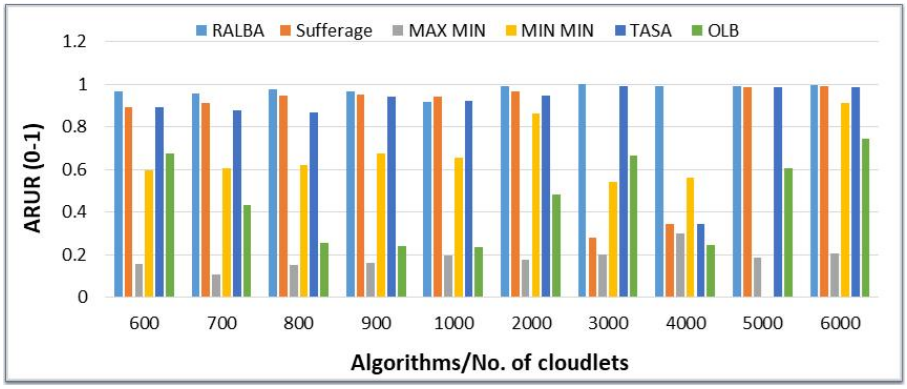

Figure 8. ARUR using GOCJ—Static algorithms.

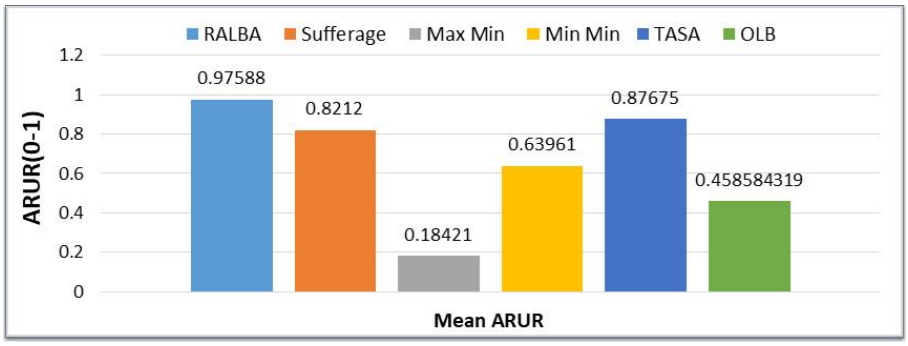

Figure 9. Mean ARUR using GOCJ—Static algorithms.

For clearer interpretation, the mean ARUR results summarized in **Figure 9** provide an aggregated view of resource utilization across static scheduling techniques, further highlighting the relative effectiveness of RALBA.

Figure 10 presents the average resource utilization ratio (ARUR) results for the dynamic scheduling algorithms. The results indicate that DE-RALBA, DRALBA, PSSLB, and PSSELB achieve higher resource utilization by efficiently exploiting the available cloud resources.

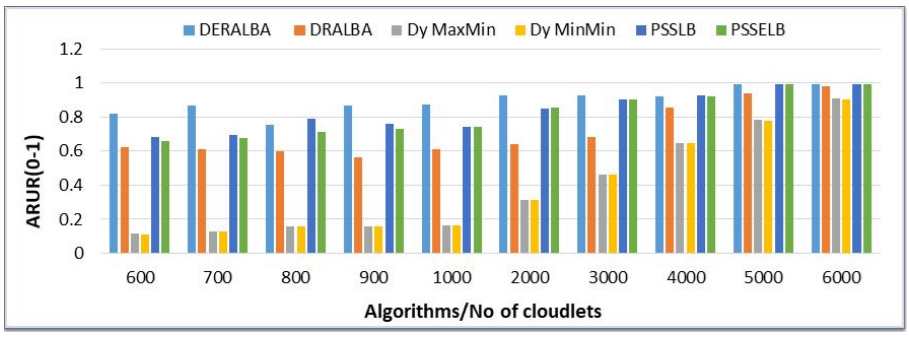

Figure 10. ARUR using GOCJ—Dynamic algorithms.

For clearer interpretation, the mean ARUR results are illustrated in **Figure 11**. Among the evaluated approaches, DE-RALBA outperforms all other dynamic algorithms, attaining a mean ARUR value of 0.894, which reflects superior and consistent resource utilization. In contrast, Dy-Max-Min and Dy-Min-Min exhibit comparatively lower ARUR values, indicating less efficient utilization of computing resources.

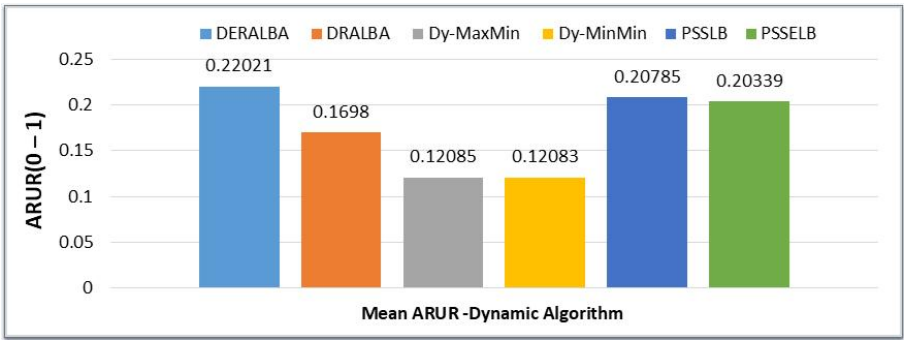

Figure 11. Mean ARUR using GOCJ—Dynamic algorithms.

3.3. Energy consumption results—GoCJ dataset

This section presents energy consumption and average energy consumption results for static scheduling algorithms. **Figure 12** compares the energy consumption behavior of static algorithms using instances from the GoCJ dataset. The results reveal that the Max–Min algorithm exhibits the highest energy consumption, rendering it the least energy-efficient approach due to ineffective resource utilization, as also reflected by its low mean ARUR values shown in **Figure 9**.

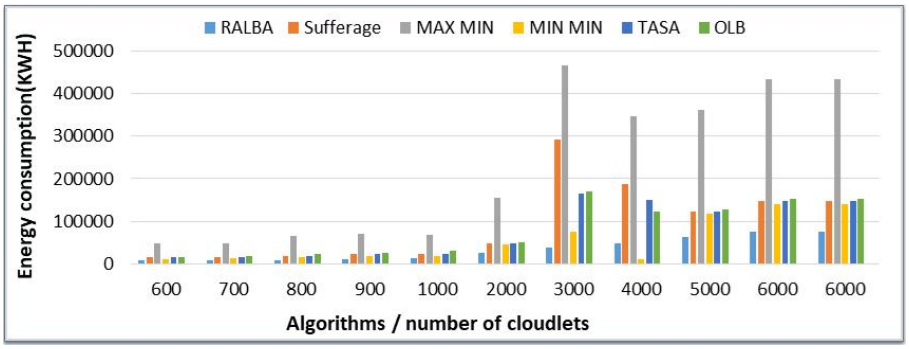

Figure 12. Energy Consumption using GOCJ—Static algorithms.

The Sufferage algorithm also incurs relatively high energy consumption but performs marginally better than Max–Min. In contrast, Min–Min, OLB, and TASA demonstrate moderate energy consumption, achieving a reasonable balance between scheduling efficiency and power usage. Notably, RALBA emerges as the most energy-efficient static scheduling algorithm, attaining the lowest energy consumption while also achieving the highest mean ARUR, as illustrated in **Figure 9**. This highlights RALBA's effectiveness in optimizing both resource utilization and energy efficiency.

For clearer interpretation, the average energy consumption of the static scheduling algorithms is illustrated in **Figure 13**. The Max–Min algorithm exhibits the highest energy consumption, with a value of 206,377.73 kWh, indicating poor energy efficiency. In contrast, RALBA records the lowest energy consumption at 29,678.12 kWh, establishing it as the most energy-efficient and optimal static scheduling technique among the evaluated algorithms.

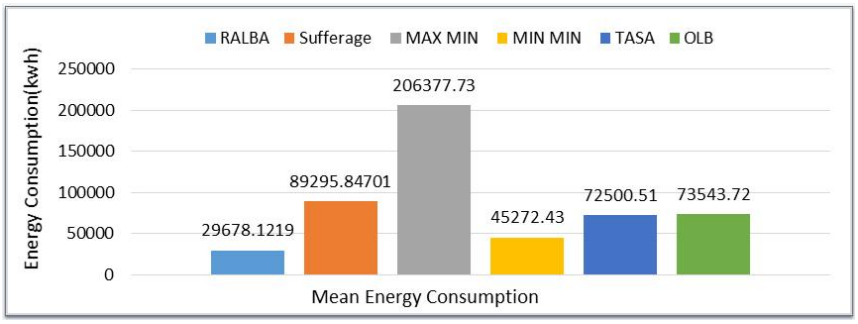

Figure 13. Average energy consumption using GOCJ—Static algorithms.

Figure 14 illustrates the total and average energy consumption of dynamic scheduling algorithms using the GoCJ dataset. Lower energy consumption indicates better energy efficiency. From the observed results, Dy-Min-Min and Dy-Max-Min consistently exhibit lower energy consumption across most cloudlet sizes, particularly for small to medium workloads, making them comparatively more energy-efficient among the evaluated dynamic approaches. In contrast, PSSLB and PSSELB demonstrate noticeably higher energy consumption, especially as the number of cloudlets increases. This suggests that, although these algorithms are designed to improve load balancing and makespan, their scheduling mechanisms introduce additional computational overhead, resulting in increased energy usage. DE-RALBA and DRALBA show moderate energy consumption, performing better than PSSLB and PSSELB but not consistently outperforming Dy-Min-Min and Dy-Max-Min. As workload size increases (e.g., 4,000–6,000 cloudlets), DE-RALBA remains relatively stable and competitive, indicating that its resource-aware strategy helps control excessive energy growth under large-scale conditions.

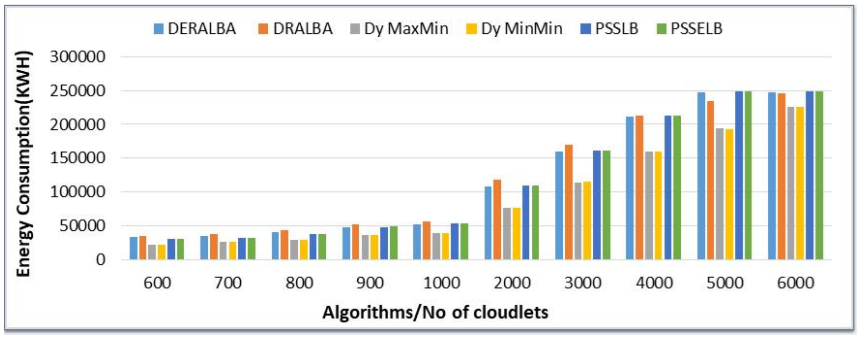

Figure 14. Energy Consumption using GOCJ—Dynamic algorithms.

Overall, the results indicate that energy efficiency in dynamic scheduling is highly dependent on the balance between scheduling overhead and task distribution efficiency. While Dy-Min-Min and Dy-Max-Min benefit from simpler decision mechanisms that reduce energy overhead, more sophisticated approaches such as PSSLB and PSSELB incur higher energy costs due to increased computational complexity. DE-RALBA provides a balanced trade-off, offering improved scalability with moderate energy consumption.

The superior performance of DE-RALBA can be attributed to its resource-aware task allocation strategy and effective load-balancing mechanism, which collectively minimize resource idleness and reduce unnecessary energy expenditure. Overall,

DE-RALBA emerges as the most effective algorithm for minimizing energy consumption, while Dy-Min-Min and Dy-Max-Min perform poorly in terms of energy efficiency.

Figure 15 presents the average energy consumption of dynamic scheduling algorithms across all instances of the GoCJ dataset. The results clearly indicate that Dy-Min-Min and Dy-Max-Min achieve the lowest average energy consumption, with values of approximately 92,162.94 kWh and 92,369.88 kWh, respectively. This confirms their relatively higher energy efficiency, as also observed in the detailed instance-level results in **Figure 14**. In contrast, PSSLB and PSSELB exhibit the highest average energy consumption, both exceeding 118,000 kWh, reflecting the additional computational overhead incurred by their more complex scheduling strategies. Although these algorithms are designed to improve load balancing and execution performance, their increased processing requirements contribute to higher overall energy usage. DE-RALBA and DRALBA demonstrate moderate average energy consumption, with values of around 118,240.35 kWh and 120,515.98 kWh, respectively. While DE-RALBA performs better than DRALBA, both approaches consume more energy compared to Dy-Min-Min and Dy-Max-Min. However, their relatively stable performance across different workload sizes suggests that they provide a balanced trade-off between adaptability and energy efficiency.

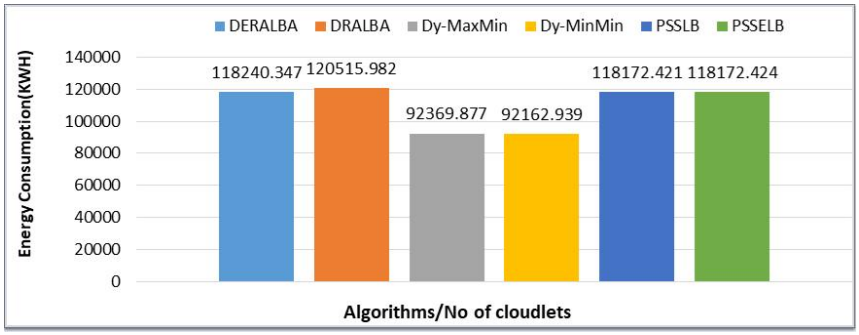

Figure 15. Average energy consumption using GOCJ—Dynamic algorithms.

Overall, the results reinforce that simpler dynamic scheduling techniques, such as Dy-Min-Min and Dy-Max-Min, tend to achieve better energy efficiency, whereas more sophisticated load-balancing approaches like PSSLB and PSSELB incur additional energy costs due to higher scheduling overhead. DE-RALBA lies between these extremes, offering moderate energy consumption while maintaining improved load distribution capabilities.

3.4. Average makespan results—HCSP dataset

This section presents the average makespan results for both static and dynamic scheduling algorithms. The evaluation is conducted using multiple instances of the HCSP workload, namely *u_c_hilo*, *u_c_lohi*, *u_c_lolo*, *u_i_hilo*, *u_i_lohi*, *u_i_lolo*, *u_s_hilo*, *u_s_lohi*, and *u_s_lolo*, which collectively represent varying levels of task consistency and heterogeneity. **Figure 16** illustrates the comparative average makespan performance of the static and dynamic scheduling algorithms across these HCSP instances.

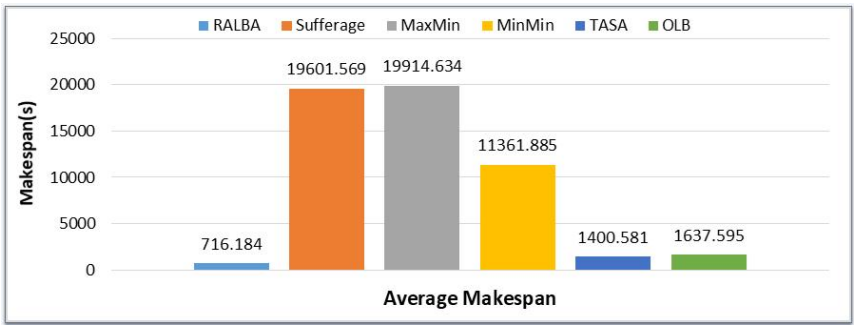

Figure 16. Average makespan using HCSP—Static algorithms.

Among the static scheduling algorithms, RALBA consistently executes the HCSP workloads with the shortest makespan, demonstrating superior time efficiency across all instances. In contrast, Sufferage and Max–Min exhibit the longest execution times, particularly for HCSP instances characterized by high workload intensity, making them less efficient in terms of execution time. Min–Min performs better than Max–Min but remains less effective than RALBA, while TASA and OLB deliver moderate performance. Overall, RALBA outperforms all other static approaches by completing HCSP instances within the least execution time.

In the case of dynamic scheduling algorithms, as illustrated in **Figure 17**, Dy-Max-Min consistently achieves the shortest makespan across all HCSP instances. Conversely, DRALBA exhibits the poorest performance, recording the longest execution times, particularly for the *u_s_lohi* instance, where its makespan exceeds 7,000,000 s. DE-RALBA ranks second among the dynamic approaches, achieving the shortest makespan after Dy-Max-Min, and thus demonstrates competitive performance in dynamically scheduling HCSP workloads.

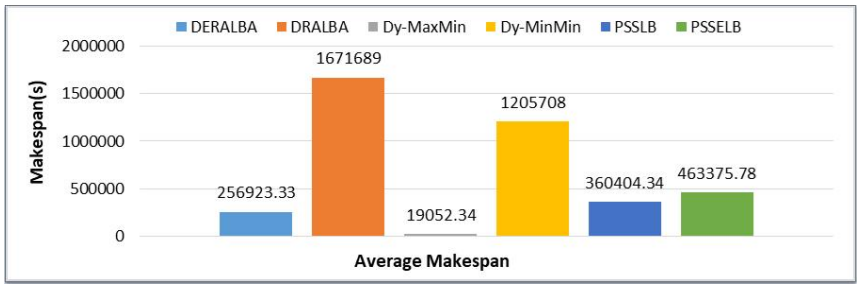

Figure 17. Average makespan using HCSP—Dynamic algorithms.

3.5. Mean ARUR results—HCSP dataset

Figures 18 and **19** present the ARUR results for both static and dynamic scheduling algorithms evaluated using HCSP dataset instances.

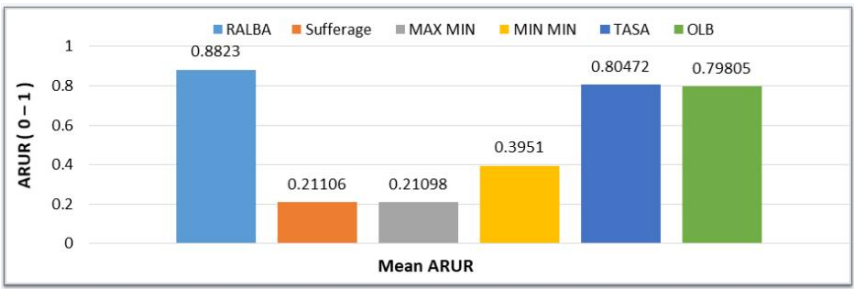

Figure 18. Mean ARUR using HCSP—Static algorithms.

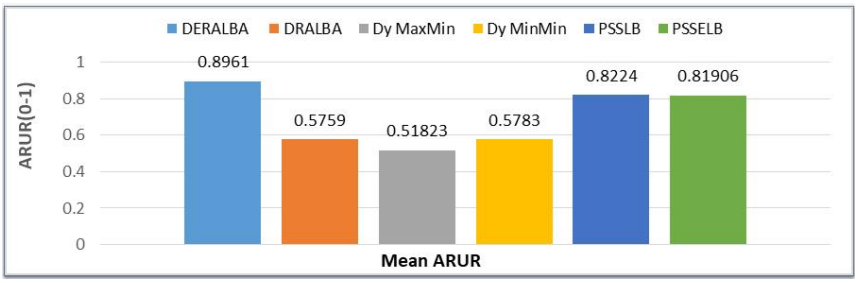

Figure 19. Mean ARUR using HCSP—Dynamic algorithms.

RALBA achieves the highest level of resource utilization across all evaluated HCSP instances, establishing it as the most efficient static scheduling algorithm. OLB also demonstrates strong performance, maintaining comparatively high resource utilization across diverse HCSP workloads. TASA exhibits moderate effectiveness, ranking between the best- and worst-performing approaches. In contrast, Sufferage and Max–Min record the lowest ARUR, indicating inefficient utilization of available resources.

For low-load HCSP instances (*u_c_lolo*, *u_i_lolo*, and *u_s_lolo*), a general decline in resource utilization is observed across all algorithms. Nevertheless, RALBA and OLB continue to outperform other static techniques by sustaining relatively higher utilization levels under reduced workload conditions. Overall, these results highlight RALBA as the most effective algorithm for maximizing resource utilization, while Sufferage and Max–Min prove to be the least effective in this regard.

Among the dynamic scheduling algorithms, DE-RALBA consistently achieves the highest ARUR across all HCSP instances, demonstrating its strong capability to effectively exploit available computing resources. PSSLB and PSSELB also yield promising results, achieving relatively high levels of resource utilization in most scenarios. In contrast, DRALBA, Dy-Max-Min, and Dy-Min-Min exhibit lower ARUR values, indicating less efficient resource utilization, with Dy-Max-Min performing the poorest in the majority of evaluated cases.

3.6. Energy consumption results—HCSP dataset

Figures 20 and **21** show the average energy consumption results of static and dynamic algorithms.

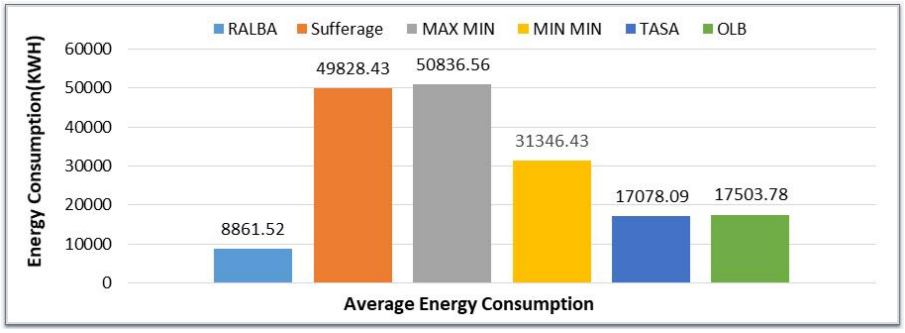

Figure 20. Average energy consumption using HCSP—Static algorithms.

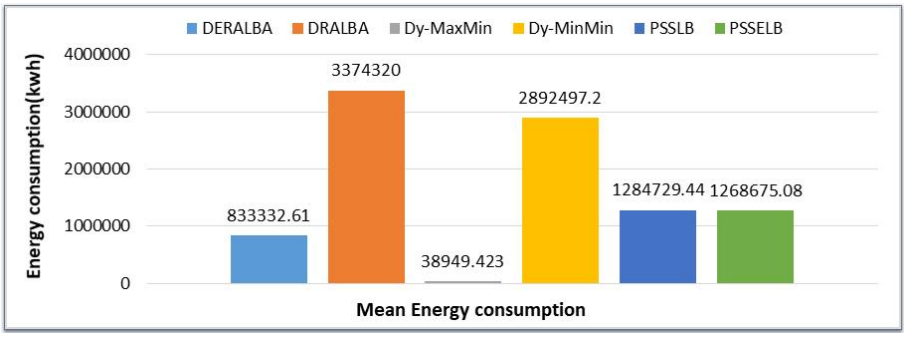

Figure 21. Average energy consumption using HSCSP—Dynamic algorithms.

Among the static scheduling algorithms, RALBA consistently remains the most energy-efficient when executing HCSP instances. In contrast, Sufferage and Max–Min exhibit significantly higher energy consumption compared to other static approaches, rendering them less effective in terms of energy efficiency. Min–Min, TASA, and OLB demonstrate moderate energy consumption, with values that are relatively close to one another, indicating a balanced trade-off between performance and energy usage. Nevertheless, Sufferage and Max–Min persist as the least favorable static algorithms with respect to energy savings.

In the case of dynamic scheduling algorithms, Dy-Max-Min consumes the least amount of energy across all HCSP instances, establishing itself as the most energy-efficient dynamic approach. DE-RALBA also performs competitively, achieving low energy consumption while executing the workload effectively. Similarly, PSSLB and PSSELB yield favorable results by maintaining relatively low energy usage. Conversely, DRALBA and Dy-Min-Min incur substantially higher energy consumption, particularly during the execution of the *u_i_lohi* and *u_s_lohi* HCSP instances, where DRALBA records the highest observed energy usage.

4. Discussion

The empirical evaluation provides comprehensive insights into the performance of cloud scheduling algorithms with respect to makespan, resource utilization, and energy consumption. The empirical evaluation indicates that RALBA and DE-RALBA demonstrate strong performance across several metrics, particularly in terms of resource utilization and load balancing. However, their effectiveness is context-dependent, and performance varies based on workload characteristics and system heterogeneity. These algorithms exhibit superior resource awareness and energy efficiency by effectively distributing workloads in a balanced manner across available cloud resources.

The strong performance of RALBA and DE-RALBA can be attributed to their intelligent task-to-resource matching mechanisms, which consider both resource capabilities and current workload conditions. By minimizing load imbalance, these approaches reduce idle times and prevent excessive resource contention, leading to lower execution times and reduced energy consumption. Furthermore, DE-RALBA's ability to dynamically adapt to workload variations enables consistent and robust performance across diverse dataset instances, highlighting its suitability for real-world

cloud environments characterized by workload dynamism and heterogeneity. In contrast, traditional heuristic-based algorithms such as Max–Min, Dy-Min-Min, and Dy-Max-Min demonstrate comparatively poor performance.

Overall, the results underscore the importance of resource-aware and load-balanced scheduling strategies in achieving optimal cloud performance. While simpler heuristics may offer lower scheduling overhead, their inability to effectively manage heterogeneous workloads and dynamically changing resource conditions limits their efficiency. In comparison, RALBA and DE-RALBA demonstrate that incorporating real-time resource awareness and adaptive load balancing is critical for reducing execution time, maximizing resource utilization, and minimizing energy consumption in modern cloud computing environments.

It is important to note that no single scheduling algorithm universally outperforms others across all scenarios. While RALBA and DE-RALBA provide advantages in resource-aware load distribution, dynamic approaches may incur higher energy consumption due to rescheduling overhead, and simpler heuristics such as Dy-Min-Min and Dy-Max-Min can achieve better energy efficiency under certain workload conditions. This highlights the inherent trade-off between adaptability, performance, and energy consumption in cloud scheduling.

5. Limitation and future work

One limitation of this study is the use of a CPU utilization-based linear power model for energy estimation, which does not explicitly account for memory and other subsystem contributions. Although prior studies indicate that CPU and memory together constitute a significant share of total data center power consumption, this work focuses on compute-centric cloud environments, where workloads are predominantly CPU-intensive, and CPU utilization is the primary driver of dynamic power consumption. Consequently, the adopted model provides a widely accepted and effective approximation for comparative evaluation of scheduling algorithms. However, this simplification may not fully capture energy variations in memory-intensive or heterogeneous workloads, where memory utilization plays a more prominent role.

In addition, the study models each job as an independent cloudlet, which does not capture task dependencies present in workflow-based applications such as MapReduce pipelines, scientific workflows, or microservice architectures. Such dependencies may introduce additional scheduling constraints, including task precedence and synchronization requirements, which are beyond the scope of the current work.

As future work, we intend to extend the proposed framework by incorporating multi-resource power models that consider both CPU and memory utilization, as well as workflow-based workloads using Directed Acyclic Graph (DAG) models (e.g., via WorkflowSim). This will enable more accurate energy estimation and a more realistic evaluation of scheduling strategies under dependency-aware cloud application scenarios.

6. Conclusion

This paper explores empirical insights on the energy consumption of cloud scheduling algorithms. It evaluates twelve popular scheduling methods, both static and dynamic, in terms of makespan, resource utilization, and energy consumption using CloudSimPlus with two scientific datasets: HCSP instances and GoCJ instances. The findings reveal that RALBA emerges as a resource-aware and energy-efficient static scheduling technique, achieving reduced makespan, enhanced resource utilization, and lower energy consumption. Conversely, the Max-Min technique exhibits the highest makespan, low throughput, poor resource utilization, and the greatest energy consumption. Among dynamic techniques, DE-RALBA demonstrates superior resource utilization and competitive makespan results while maintaining relatively efficient energy consumption. Nonetheless, dynamic scheduling methods generally exhibit higher energy consumption than static techniques due to the frequent adjustments required for job scheduling.

Author contributions: Conceptualization, SN and AH; methodology, SN and AH; software, SN; validation, SN and AH; formal analysis, SN; investigation, SN; data curation, SN and AH; writing—original draft preparation, SN and AH; writing—review and editing, AH; visualization, SN and AH; supervision, AH; project administration, AH. All authors have read and agreed to the published version of the manuscript.

Funding: This work received no external funding.

Institutional review board statement: Not applicable.

Informed consent statement: Not applicable.

Data availability statement: Not applicable.

Conflict of interest: The authors declare no conflict of interest.

AI use statement: During the preparation of this manuscript, the authors used Microsoft Copilot to assist with grammar correction, proofreading, and manuscript formatting. The authors carefully reviewed and revised all AI-assisted outputs and take full responsibility for the accuracy, integrity, and final content of the manuscript.

References

1. Atiewi S, Yussof S. Comparison between Cloud Sim and Green Cloud in Measuring Energy Consumption in a Cloud Environment. In: Proceedings of the 2014 3rd International Conference on Advanced Computer Science Applications and Technologies; 29–30 December 2014; Amman, Jordan. pp. 9–14. doi: 10.1109/ACSAT.2014.9
2. Mell PM, Grance T. The NIST definition of cloud computing (No. NIST SP 800-145). National Institute of Standards and Technology; 2011. doi: 10.6028/NIST.SP.800-145
3. Atiewi S, Abuhussein A, Saleh MA. Impact of Virtualization on Cloud Computing Energy Consumption: Empirical Study. In: Proceedings of the ISCSIC'18: The 2nd International Symposium on Computer Science and Intelligent Control; 21–23 September 2018; Stockholm, Sweden. pp. 1–7. doi: 10.1145/3284557.3284738
4. Hussain A, Aleem M, Islam MA, et al. A Rigorous Evaluation of State-of-the-Art Scheduling Algorithms for Cloud Computing. IEEE Access. 2018; 6: 75033–75047. doi: 10.1109/ACCESS.2018.2884480
5. Ismail L, Fardoun A. EATS: Energy-Aware Tasks Scheduling in Cloud Computing Systems. Procedia Computer

Science. 2016; 83: 870–877. doi: 10.1016/j.procs.2016.04.178
6. Murad SA, Muzahid AJM, Azmi ZRM, et al. A review on job scheduling technique in cloud computing and priority rule based intelligent framework. Journal of King Saud University - Computer and Information Sciences. 2022; 34(6): 2309–2331. doi: 10.1016/j.jksuci.2022.03.027
7. Shaik N, Jyotheeswai P. A survey on energy aware job scheduling algorithms in cloud environment. i-manager's Journal on Cloud Computing. 2016; 3(1): 30.
8. Chen F, Schneider JG, Yang Y, et al. An energy consumption model and analysis tool for Cloud computing environments. In: Proceedings of the 2012 First International Workshop on Green and Sustainable Software (GREENS); 3 June 2012; Zurich, Switzerland. pp. 45–50. doi: 10.1109/GREENS.2012.6224255
9. Beloglazov A, Buyya R. 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. 2012; 24(13): 1397–1420. doi: 10.1002/cpe.1867
10. Berl A, Gelenbe E, Di Girolamo M, et al. Energy-Efficient Cloud Computing. The Computer Journal. 2010; 53(7): 1045–1051. doi: 10.1093/comjnl/bxp080
11. Panwar SS, Rauthan MMS, Barthwal V. A systematic review on effective energy utilization management strategies in cloud data centers. Journal of Cloud Computing. 2022; 11(1): 95. doi: 10.1186/s13677-022-00368-5
12. Hussain A, Aleem M. GoCJ: Google Cloud Jobs Dataset for Distributed and Cloud Computing Infrastructures. Data. 2018; 3(4): 38. doi: 10.3390/data3040038
13. Braun TD, Siegel HJ, Beck N, et al. A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. Journal of Parallel and Distributed Computing. 2001; 61(6): 810–837. doi: 10.1006/jpdc.2000.1714
14. Khair Y, Benlabbes H. Opportunistic Load Balancing for Virtual Machines Scheduling in a Cloud Environment. Engineering Proceedings. 2023; 29(1): 1. doi: 10.3390/engproc2023029001
15. Qaisar MUF, Wang X, Hawbani A, et al. TORAS: Trustworthy Load-Balanced Opportunistic Routing for Asynchronous Duty-Cycled WSNs. IEEE Systems Journal. 2023; 17(2): 2259–2270. doi: 10.1109/JSYST.2022.3221096
16. Rao PH, Rajakumar PS, Geetha S. Load Balancing Issues and Challenges in Cloud Environment. In: Proceedings of the 2024 International Conference on Electronic Systems and Intelligent Computing (ICESIC); 22–23 November 2024; Chennai, India. pp. 1–6. doi: 10.1109/ICESIC61777.2024.10846533
17. Ali SA, Alam M. A relative study of task scheduling algorithms in cloud computing environment. In: Proceedings of the 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I); 14–17 December 2016; Greater Noida, India. pp. 105–111. doi: 10.1109/IC3I.2016.7917943
18. Patni S, Saxena D, Singh AK. Resource Management in Cloud Computing: Concepts and Implementation. Springer Nature; 2025. doi: 10.1007/978-3-031-83053-2
19. Huankai Chen, Wang F, Helian N, et al. User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing. In: Proceedings of the 2013 National Conference on Parallel Computing Technologies (PARCOMPTECH); 21–23 February 2013; Bangalore, India. pp. 1–8. doi: 10.1109/ParCompTech.2013.6621389
20. Maipan-Uku, JY, Muhammed A, Abdullah A, et al. Efficient Loads Balancing for Grid Computing System Using Min-Min Scheduling Algorithm. International Journal of Applied Engineering Research (IJAER). 2015; 10(95): 75–79
21. Mathew T, Sekaran KC, Jose J. Study and analysis of various task scheduling algorithms in the cloud computing environment. In: Proceedings of the 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI); 24–27 September 2014; Delhi, India. pp. 658–664. doi: 10.1109/ICACCI.2014.6968517
22. Parsa S, Entezari-Maleki R. RASA: A New Grid Task Scheduling Algorithm. International Journal of Digital Content Technology and its Applications. 2009; 3(4): 91–99. Available online: https://www.researchgate.net/publication/220670335_RASA_a_new_grid_task_scheduling_algorithm
23. Raeisi-Varzaneh M, Dakkak O, Fazea Y, et al. Advanced cost-aware Max–Min workflow tasks allocation and scheduling in cloud computing systems. Cluster Computing. 2024; 27(9): 13407–13419. doi: 10.1007/s10586-024-04594-1
24. Ninh A, Shen ZJM. Stochastic Resource Allocation Problems: Minmax and Maxmin Solutions. Manufacturing & Service Operations Management. 2024; 26(6): 2322–2335. doi: 10.1287/msom.2022.0550
25. Murad SA, Azmi ZRM, Muzahid AJM, et al. SG-PBFS: Shortest Gap-Priority Based Fair Scheduling technique for job scheduling in cloud environment. Future Generation Computer Systems. 2024; 150: 232–242. doi:

10.1016/j.future.2023.09.005
26. Tsai HC, Hong YWP, Sheu JP. Completion Time Minimization for UAV-Enabled Surveillance Over Multiple Restricted Regions. IEEE Transactions on Mobile Computing. 2022; 1–14. doi: 10.1109/TMC.2022.3200732
27. Sino M, Domazet E. Scalable Parallel Processing: Architectural Models, Real-Time Programming, and Performance Evaluation. Engineering Proceedings. 2025; 104(1): 60. doi: 10.3390/engproc2025104060
28. Hussain A, Aleem M, Khan A, et al. RALBA: a computation-aware load balancing scheduler for cloud computing. Cluster Computing. 2018; 21(3): 1667–1680. doi: 10.1007/s10586-018-2414-6
29. Awan A, Aleem M, Hussain A, et al. DFARM: a deadline-aware fault-tolerant scheduler for cloud computing. Cluster Computing. 2024; 27(7): 9323–9344. doi: 10.1007/s10586-024-04419-1
30. Mao Y, Chen X, Li X. Max–Min Task Scheduling Algorithm for Load Balance in Cloud Computing. In: Proceedings of International Conference on Computer Science and Information Technology, Advances in Intelligent Systems and Computing; 27–29 October 2025; Paris, France. pp. 457–465. doi: 10.1007/978-81-322-1759-6_53
31. Alaei N, Safi-Esfahani F. RePro-Active: a reactive–proactive scheduling method based on simulation in cloud computing. The Journal of Supercomputing. 2018; 74(2): 801–829. doi: 10.1007/s11227-017-2161-0
32. Nabi S, Ibrahim M, Jimenez JM. DRALBA: Dynamic and Resource Aware Load Balanced Scheduling Approach for Cloud Computing. IEEE Access. 2021; 9: 61283–61297. doi: 10.1109/ACCESS.2021.3074145
33. Hussain A, Aleem M, Ur Rehman A, et al. DE-RALBA: dynamic enhanced resource aware load balancing algorithm for cloud computing. PeerJ Computer Science. 2025; 11: e2739. doi: 10.7717/peerj-cs.2739
34. Khan ZI, Khan M, Shah SNM. MAADRR: Multi-Agent Adaptive Dynamic Round Robin for Cloud Scheduling Using Real Cloud Workload Traces. IEEE Access. 2026; 14: 44165–44184. doi: 10.1109/ACCESS.2026.3676311
35. Guérout T, Monteil T, Da Costa G, et al. Energy-aware simulation with DVFS. Simulation Modelling Practice and Theory. 2013; 39: 76–91. doi: 10.1016/j.simpat.2013.04.007
36. Minas L, Ellison B. Energy Efficiency for Information Technology: How To Reduce Power Consumption in Servers and Data Centers. Intel Press; 2009.
37. Beloglazov A, Abawajy J, Buyya R. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. Future Generation Computer Systems. 2012; 28(5): 755–768. doi: 10.1016/j.future.2011.04.017
38. Arshad U, Aleem M, Srivastava G, et al. Utilizing power consumption and SLA violations using dynamic VM consolidation in cloud data centers. Renewable and Sustainable Energy Reviews. 2022; 167: 112782. doi: 10.1016/j.rser.2022.112782
39. Ahvar E, Orgerie AC, Lebre A. Estimating Energy Consumption of Cloud, Fog, and Edge Computing Infrastructures. IEEE Transactions on Sustainable Computing. 2022; 7(2): 277–288. doi: 10.1109/TSUSC.2019.2905900
40. Lin W, Wang H, Zhang Y, et al. A cloud server energy consumption measurement system for heterogeneous cloud environments. Information Sciences. 2018; 468: 47–62. doi: 10.1016/j.ins.2018.08.032
41. Beloglazov A, Buyya R, Lee YC, et al. A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems. In: Advances in Computers. Elsevier; 2011. pp. 47–111. doi: 10.1016/B978-0-12-385512-1.00003-7
42. Liu Z, Cho S. Characterizing Machines and Workloads on a Google Cluster. In: Proceedings of the 2012 41st International Conference on Parallel Processing Workshops; 10–13 September 2012; Pittsburgh, PA, USA. pp. 397–403. doi: 10.1109/ICPPW.2012.57
43. Moreno IS, Garraghan P, Townend P, et al. An Approach for Characterizing Workloads in Google Cloud to Derive Realistic Resource Utilization Models. In: Proceedings of the 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering; 25–28 March 2013; San Francisco, CA, USA. pp. 49–60. doi: 10.1109/SOSE.2013.24
44. Reiss C, Tumanov A, Gregory GR, et al. Towards Understanding Heterogeneous Clouds At Scale: Google Trace Analysis. Intel Science and Technology Center for Cloud Computing; 2012. Available online: https://istc-cc.cmu.edu/publications/papers/2012/ISTC-CC-TR-12-101.pdf
45. Kavulya S, Tan J, Gandhi R, et al. An Analysis of Traces from a Production MapReduce Cluster. In: Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing; 17–20 May 2010; Melbourne, VIC, Australia. pp. 94–103. doi: 10.1109/CCGRID.2010.112
46. Gad AG, Houssein EH, Zhou M, et al. Damping-Assisted Evolutionary Swarm Intelligence for Industrial IoT Task Scheduling in Cloud Computing. IEEE Internet of Things Journal. 2024; 11(1): 1698–1710. doi: 10.1109/JIOT.2023.3291367

47. Choppara P, Mangalampalli SS. A Hybrid Task Scheduling Technique in Fog Computing Using Fuzzy Logic and Deep Reinforcement Learning. IEEE Access. 2024; 12: 176363–176388. doi: 10.1109/ACCESS.2024.3505546
48. Choppara P, Mangalampalli SS. Reliability and Trust Aware Task Scheduler for Cloud-Fog Computing Using Advantage Actor Critic (A2C) Algorithm. IEEE Access. 2024; 12: 102126–102145. doi: 10.1109/ACCESS.2024.3432642