*I.J. Intelligent Systems and Applications,* **2026, 4, 73-96**
Published Online on August 8, 2026 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2026.04.05

# Energy and Carbon Emission Aware Task Scheduling in Cloud Computing Using Memetic Optimization Framework

**Chennoji Sandhya**
Geethanjali College of Engineering and Technology, Hyderabad, Telangana, India
E-mail: Sandhya.chennoji@gmail.com
ORCID iD: https://orcid.org/0009-0003-7202-7130

**Mandla Alphonsa**
Vasavi college of Engineering, Hyderabad, Telangana, India
E-mail: alphonsa.mandla@staff.vce.ac.in
ORCID iD: https://orcid.org/0009-0007-6741-1516

**Vankudoth Biksham**
School of Engineering, Department of Data Science, Anurag University, Hyderabad, Telangana, India
E-mail: vbm2k2@gmail.com
ORCID iD: https://orcid.org/0000-0003-0532-594X

**T. L. Deepika Roy**
Koneru Lakshmaiah Education Foundation, Green fields, Guntur, Andhra Pradesh, India
E-mail: thotadeepika001@gmail.com
ORCID iD: https://orcid.org/0009-0004-4851-2563

**Santhosh Kumar Medishetti***
Nalla Narasimha Reddy Education Society's Group of Institutions, Hyderabad, Telangana, India
E-mail: medishettysantosh@gmail.com
ORCID iD: https://orcid.org/0000-0001-5936-6582
*Corresponding author

Received: January 8, 2026; Revised: March 15, 2026; Accepted: June 21, 2026; Published: August 8, 2026

**Abstract:** Minimizing energy consumption and carbon emissions while maintaining system performance is a critical challenge in cloud task scheduling. This paper presents a multi-objective scheduling framework based on a Memetic Algorithm (MA) designed to optimize task-to-VM mapping with respect to energy efficiency, carbon footprint, and throughput. The algorithm employs a weighted fitness function that integrates actual and idle energy usage, simulated time-varying carbon intensity, and task throughput. To enhance solution quality, MA combines global evolutionary operations (selection, crossover, mutation) with local search heuristics that adaptively refine candidate solutions based on workload characteristics and green energy opportunities. The carbon emission model incorporates dynamic emission factors (γ) derived from location- and time-sensitive datasets, reflecting real-world variability in grid carbon intensity. The proposed method is evaluated using the NASA Ames iPSC/860 workload under both low and high resource utilization scenarios. Comparative results demonstrate that the proposed MA approach achieves a 20.2% reduction in carbon emissions, reduces energy consumption by 17.7%, and enhances throughput by 21.2% over conventional techniques such as HDDPGTS and RAPTS, while also ensuring competitive performance in terms of makespan and resource utilization. These improvements underscore the potential of memetic-based hybrid scheduling to support environmentally sustainable and performance-efficient cloud infrastructures. The findings highlight the importance of integrating eco-aware intelligence into task scheduling policies, particularly for mission-critical and energy-intensive cloud applications.

**Index Terms:** Cloud Computing, Energy Consumption, Carbon Emission, Memetic Algorithm, Task Scheduling, Virtual Machines.

## 1. Introduction

Cloud computing now provides the foundation for modern business change through its scalable, flexible, and cost-effective services for various sectors. Enterprises, governments, and individuals leverage cloud platforms to support critical operations, ranging from data analytics and artificial intelligence to remote education and healthcare. While this paradigm has significantly enhanced computational efficiency and agility, it has simultaneously escalated the demand for computational infrastructure [1]. The growing dependence on data centers to fulfill computational requests leads to an unprecedented increase in energy usage and a corresponding rise in carbon emissions, presenting a paradox between technological advancement and environmental sustainability [2].

Data centres, as the operational core of cloud computing, are power-intensive facilities that require uninterrupted energy to operate servers, storage systems, networking equipment, and cooling infrastructure. A typical data center consumes megawatts of electricity, a substantial portion of which is used inefficiently due to poor resource utilization or idle components [3-4]. This energy consumption translates into significant carbon emissions, particularly when electricity is sourced from fossil-fuel-based grids. Reports from the Uptime Institute and the International Energy Agency (IEA) forecast that without adequate mitigation, data center energy consumption and emissions will become increasingly unsustainable, threatening global climate commitments [5].

In response to these ecological concerns, the concept of green cloud computing has gained momentum. Green computing refers to the environmentally responsible and energy-efficient use of computing resources. It focuses on optimizing energy usage, reducing carbon footprints, and maximizing resource efficiency without compromising service quality. Achieving these objectives requires innovations in hardware design, thermal management, and most crucially intelligent software-level mechanisms such as energy-aware task scheduling and resource allocation. Effective task scheduling strategies are pivotal, as they determine the operational workload distribution across available computing resources.

TS in CC involves assigning a series of tasks, each with specific computational demands, to available virtual machines (VMs) in a way that optimizes predefined objectives [6]. Traditional scheduling algorithms often emphasize system performance metrics such as makespan, load balancing, and throughput. However, in the context of green computing, scheduling must also consider energy efficiency and carbon emission reductions. This transformation of task scheduling from a performance-centric to an energy-aware multi-objective optimization problem presents unique computational challenges, especially under dynamic and heterogeneous workload conditions.

Heuristic algorithms such as Round-Robin, First-Fit, and Min-Min are widely adopted in cloud computing due to their low computational complexity, typically operating in linear time, and their ability to make fast scheduling decisions in real-time environments. These approaches are effective in scenarios where simplicity, speed, and scalability are paramount [7]. However, their decision-making is often limited to single-objective optimization or basic load-balancing criteria, which restricts their performance in more complex, heterogeneous cloud environments. In such contexts where multiple conflicting objectives like minimizing energy consumption, reducing carbon emissions, and maximizing throughput must be considered simultaneously, traditional heuristics may not provide sufficiently optimal solutions. This is where metaheuristic techniques like the Memetic Algorithm demonstrate their value. By integrating global exploration strategies (e.g., evolutionary operations) with local refinement (e.g., hill-climbing), Memetic Algorithms can navigate vast solution spaces and adapt task mappings dynamically. While they introduce a moderate computational overhead compared to simple heuristics, this is justified by their ability to achieve substantially better optimization outcomes, making them highly suitable for sustainability-focused and performance-sensitive cloud scheduling applications.

To overcome the limitations of these approaches, this research introduces a Memetic Algorithm (MA)-based task scheduling technique tailored to the dual objectives of reducing carbon emissions and optimizing energy consumption [8]. Memetic Algorithms are an advanced class of evolutionary algorithms inspired by the concept of memes in cultural evolution—units of information that undergo local adaptation and refinement. By integrating global search methods with domain-specific local search procedures, MAs can effectively navigate large, multidimensional solution spaces while ensuring fine-grained solution improvements. This dual capability is particularly advantageous in cloud environments, where energy-aware decisions require both broad exploration and precise adaptation.

The hybrid nature of Memetic Algorithms offers several operational benefits in energy-aware task scheduling. Firstly, MAs can escape local optima more effectively than conventional heuristics by combining multiple learning strategies. Secondly, the embedded local search component enables the fine-tuning of task-resource mappings based on real-time feedback, leading to better energy savings and carbon reductions [9]. Thirdly, MAs maintain high solution diversity, making them robust against the unpredictability and heterogeneity of cloud workloads. These properties position MAs as a powerful framework for balancing trade-offs between execution efficiency and environmental sustainability.

A distinguishing feature of this study is the explicit integration of carbon intensity into the scheduling model. Carbon intensity refers to the amount of carbon dioxide emitted per unit of electricity consumed. As the energy mix of power grids varies temporally and geographically, considering real-time carbon intensity data allows the scheduling algorithm to prioritize task allocation to VMs or data centers that are operating under greener power conditions. This enhances the eco-efficiency of the cloud system and aligns with regulatory and corporate carbon neutrality goals.

Despite increasing academic and industrial focus on green computing, there remains a critical gap in task scheduling strategies that jointly address energy efficiency and carbon footprint reduction within real-time constraints. Most existing

algorithms either prioritize performance at the cost of environmental impact or lack adaptability in fluctuating operational conditions. There is a pressing need for a unified, intelligent, and adaptive scheduling approach that optimally balances these competing objectives.

This study develops and tests a Memetic Algorithm-based task scheduling framework that reduces energy usage and carbon output while delivering adequate system performance. Through experiments with NASA Ames iPSC/860 workloads and VMs with low resource configurations [10], the evaluation established a robust testing platform. Our testing used carbon emission and energy consumption measures along with throughput results to demonstrate the superior performance of the proposed model compared to the HDDPGTS and RAPTS methods.

To ensure a comprehensive and realistic evaluation, the NASA Ames iPSC/860 workload trace is employed. This dataset, curated from a real-world parallel computing environment, simulates heterogeneous tasks with varying resource demands and execution times. It is widely used in cloud research due to its detailed time-stamped job records and complexity, which make it ideal for testing algorithm scalability, adaptability, and efficiency under realistic conditions. The proposed MA-based scheduler is compared against two advanced scheduling models: HDDPGTS and RAPTS. HDDPGTS leverages deep reinforcement learning for intelligent decision-making, while RAPTS emphasizes static priority-based resource allocation. Despite their strengths, both models exhibit limitations in handling real-time carbon data and optimizing local scheduling refinements. Experimental results reveal that MA outperforms these models by achieving up to 26.3% reduction in energy usage and 28.7% lower carbon emissions, demonstrating its superiority in sustainable task allocation. This research makes several noteworthy contributions to the field of sustainable CC.

- We propose a novel multi-objective TS model that simultaneously optimizes energy consumption and carbon emissions in CC environments.
- We employ a Memetic Algorithm to enhance task-to-VM mapping efficiency, enabling fine-tuned local optimization under dynamic workload conditions.
- We develop a context-aware scheduling framework that adapts to real-time fluctuations in energy usage and carbon intensity, ensuring intelligent and sustainable task allocation.
- We validate the proposed model using the NASA Ames iPSC/860 workload and benchmarked it against HDDPGTS and RAPTS, demonstrating superior performance in terms of energy and carbon optimization.

The remainder of this paper is organized into five key sections. Section 2 provides an extensive review of related research in task scheduling and optimization in cloud environments. Section 3 details the architecture and operational mechanisms of the proposed Memetic Algorithm. Section 4 outlines the experimental setup, including dataset description, parameter settings, and evaluation metrics. It also discusses the results, performance comparisons, and analytical insights. Lastly, Section 5 concludes the paper and provides future research directions. Table 1 below depicts the acronyms used in this study.

Table 1. Description of acronyms

| Notation | Description |
|---|---|
| *TS* | Task Scheduling |
| *GA* | Genetic Algorithm |
| *CC* | Cloud Computing |
| *ACO* | Ant Colony Optimization |
| *FC* | Fog Computing |
| *FCFS* | First Come First Serve |
| *RAPTS* | Resource-Aware Prioritized Task Scheduling |
| *MA* | Memetic Algorithm |
| *PSO* | Particle Swarm Optimization |
| *RR* | Round Robin |
| *VM* | Virtual Machine |

## 2. Related Works

Section 2 offers a complete examination of previous studies on cloud computing task scheduling that considers various metrics such as makespan, execution time, energy usage, carbon output, and throughput. It highlights the optimization techniques employed in previous studies and critically analyses their strengths and limitations. Additionally, a detailed comparison table is included, summarizing the key parameters, simulation environments, and datasets used.

Marri and Rajalakshmi (2022) [11] introduced MOEAGAC, a hybrid scheduling model that integrates a genetic algorithm with an energy-aware framework to optimize makespan, energy consumption, and data transfer time in cloud environments. By incorporating CPU voltage and frequency scaling into the fitness function, the model effectively reduces energy usage while maintaining performance. The approach demonstrates notable improvements over traditional

algorithms like MODPSO and HEFT, highlighting the potential of combining evolutionary strategies with energy considerations in task scheduling. Another study by Hanafy et al. (2023) [12] proposed Carbonscaler, a scheduling framework that leverages the elasticity of batch workloads to minimize carbon emissions in cloud computing. By dynamically adjusting server allocations based on real-time carbon intensity data, Carbonscaler achieves significant carbon savings compared to traditional suspend-resume and static scaling policies. Implemented within Kubernetes, the framework demonstrates the feasibility of integrating carbon-awareness into cloud resource management.

Tuli et al. (2021) [13] created HUNTER, an AI-based energy optimization tool that uses Gated Graph Convolution Networks to manage data center resources. HUNTER creates a multi-objective scheduling challenge that aligns performance and sustainability targets by integrating energy optimization with thermal and cooling physics. The HUNTER system demonstrated superior performance than traditional solutions because it reduced cloud energy use while meeting SLA targets at lower operational costs. Another similar work by Chhabra et al. (2022) [14] developed h-DEWOA by combining Whale Optimization Algorithm with both differential evolution and opposition-based learning to improve cloud task scheduling results. Their solution reduces task completion times and ensures lower energy usage for groups of parallel tasks. Real cloud workloads show h-DEWOA delivers better energy efficiency and task handling performance than conventional WOA-based algorithms and metaheuristic alternatives. Additionally, Chhabra et al. (2022) [15] created a hybrid scheduling algorithm of PSO with CS to minimize cloud task scheduling power use. The algorithm sets task and VM priorities through electricity cost to reduce both processing time and electrical power usage. Their study shows that CloudSim proves the hybrid approach outperforms standard ACO, GA, PSO, and CS algorithms by highlighting how hybrid metaheuristic methods improve energy-aware scheduling.

Bindu et al. (2018) [16] developed a multi-objective genetic algorithm focusing on energy consumption in cloud task scheduling. Unlike previous works that primarily targeted time and cost optimization, this approach integrates energy considerations into the scheduling process. Simulation results validate the algorithm's efficiency in reducing energy usage, highlighting the importance of incorporating energy metrics into multi-objective optimization frameworks. A recent work on Lyapunov optimization by Yang et al. (2022) [17] designed a scheduling system that uses carbon-intensity criteria to determine cloud network task placement for minimum carbon output. The algorithm works with a system model that uses virtual queues to track renewable power sources and optimizes performance through Lyapunov optimization methods. Their system demonstrates that AI model training tasks can save 54% of total carbon emissions when using carbon-aware scheduling compared to traditional methods on actual data sets.

Kim et al. (2023) [18] introduced Greenscale, a framework designed to optimize carbon efficiency in edge-cloud infrastructures. By considering the time and location-based carbon intensity of energy sources, Greenscale makes informed decisions on when and where to execute applications. Evaluations across AI, gaming, and AR/VR applications demonstrate up to 29.1% reduction in carbon emissions, highlighting the significance of carbon-aware scheduling in edge computing scenarios. Another study by Liu et al. (2020) [19] addressed the challenge of energy optimization under time constraints in heterogeneous cloud environments. By implementing a hardware-software collaborative strategy that includes DVFS-capable infrastructure and a Q-learning-based scheduling algorithm, the approach aims to minimize energy costs while meeting deadline requirements. The integration of Rapid Local Convolution Optimization further enhances convergence speed, making the solution suitable for large-scale datacenters.

Materwala and Ismail (2021) [20] proposed a bi-objective evolutionary algorithm that simultaneously optimizes performance and energy consumption in cloud data centers. By tracking performance counters, the algorithm finds the right balance between faster processing and lower power consumption. Experimental tests prove that their approach works better and uses less energy than current scheduling methods in the field while showing the importance of optimizing multiple cloud tasks at once. A recent work related to carbon emission done by Souza et al. (2024) [21] proposed Casper, a carbon-aware scheduling and provisioning framework for distributed web services. Casper formulates a multi-objective optimization that dynamically shifts web requests among geo-replicated clouds based on real-time carbon intensity and latency constraints. By exploiting spatial and temporal flexibility of workloads, Casper directs tasks to regions where low-carbon energy is available. Evaluations show that Casper can reduce carbon emissions by up to ~70% compared to baseline methods, with no degradation in latency or service quality. This demonstrates the effectiveness of cross-data-center replication and workload shifting in cutting cloud carbon footprints.

Xu et al. (2024) [22] address energy-efficient flow-shop scheduling with carbon considerations. They study a DHFSSP where energy consumption (and hence carbon) is minimized alongside makespan. To solve this, they develop a KDM algorithm that combines a collaborative population initialization with specialized update and local-search strategies tailored to the problem structure. In particular, KDMA includes a carbon-reduction strategy in its local search to bias solutions toward low-emission schedules. Simulation results indicate KDMA outperforms conventional heuristics and metaheuristics, yielding schedules with both shorter makespan and significantly lower total carbon output.

Miao et al. (2024) [23] propose ECMR, an *Energy and Carbon-aware scheduling* algorithm for geo-distributed machine learning workloads. ECMR targets delay-tolerant DML tasks across multiple cloud regions powered by heterogeneous renewable mixes. It leverages spatiotemporal complementarity of wind and solar generation, assigning tasks to data centers when and where renewable energy is abundant. Compared to baseline schedulers, ECMR substantially cuts total power usage, energy costs, and CO2 emissions of the cloud system. Notably, the algorithm boosts renewable utilization (to ~90.8%) while maintaining acceptable QoS, and achieves fast response times (~12.6ms average) and low failure rates, demonstrating its practical efficacy in green scheduling.

Gu et al. (2018) [24] introduced GAIA, a carbon-aware batch-job scheduler that explicitly trades off performance,

cost, and carbon emissions. GAIA operates on a cloud platform (e.g., AWS) and uses different procurement options (on-demand, reserved, spot instances) together with temporal workload shifting. It taps into real-time carbon intensity signals and supports flexible job queues to decide *when* and *where* to run each batch job. The study shows GAIA can double the carbon savings per percentage cost increase compared to prior policies, while also reducing the performance overhead by about 26%. In other words, GAIA achieves roughly twice the carbon reduction (for a given budget) relative to state-of-the-art carbon-aware schedulers, validating its hybrid scheduling strategy for greener clouds.

Mikram et al. (2024) [25] presented a two-step hybrid scheduling method (GAECS) that is both energy- and time-aware. In the first step, tasks are prioritized and a genetic algorithm generates initial schedules; in the second step, an *energy-conscious scheduling heuristic* assigns tasks to processors. This hybrid GA-heuristic approach is designed to jointly minimize makespan and energy usage. Simulation results indicate that GAECS outperforms benchmark algorithms across various metrics, consistently delivering lower execution time and power consumption for cloud task workloads.

Zhao et al. (2024) [26] proposed ERLFC, a federated-cloud scheduling framework using reinforcement learning to reduce energy and carbon. ERLFC uses an actor-critic agent that observes the state of each geographically-distributed data center (including its current load, energy mix, cooling method, and grid carbon intensity) to assign incoming tasks. By learning from real-world task traces, ERLFC effectively exploits regional differences in carbon intensity. Experimental comparisons with heuristics (Round Robin, Greedy, etc.) show that ERLFC significantly cuts both energy consumption and CO2 emissions. Specifically, it achieves roughly 1.1–1.3× greater energy and carbon reduction than conventional schedulers, demonstrating that RL can capture complex trade-offs for greener cloud operation.

Guizzo et al. (2017) [27] introduced HHCSP, a *multi-armed bandit hyper-heuristic* for multi-objective cloud scheduling. HHCSP treats the selection among different scheduling heuristics as a bandit problem, extending the UCB algorithm with a fitness-rate ranking mechanism. The goal is to jointly optimize makespan, execution cost, and carbon emissions for cloud tasks. Evaluated on real AWS workloads, HHCSP outperforms state-of-the-art multi-objective schedulers. It achieves a superior Pareto front (as measured by Hypervolume and IGD+) relative to baselines, yielding lower carbon and cost for given performance targets. This demonstrates the promise of adaptive, hybrid metaheuristics for sustainable cloud scheduling.

Abbasi-Khazaei et al. (2022) [28] compared genetic and memetic algorithms for dynamic VM placement in energy-heterogeneous clouds. Their model considers multiple energy sources (renewable vs. brown) feeding the data center, and prioritizes renewable usage to cut carbon. The evolutionary solver reassigns VMs over time to match VM load to low-carbon periods. According to their reported results, the proposed GA and memetic placement schemes not only reduce server idle time but also significantly lower overall energy draw and $CO_2$ costs by exploiting renewable availability. This work highlights how hybrid evolutionary strategies can efficiently optimize VM allocation for green objectives under realistic power supply scenarios.

Xiao et al. (2024) [29] developed FTL ("Follow-The-Leader"), a meta-scheduling strategy for delay-tolerant batch jobs that continually adapts to minimize carbon. FTL dynamically monitors job and grid characteristics and selects among pre-defined carbon-aware policies based on recent performance (inspired by Follow-the-Leader online learning). It thus automatically tunes its scheduling behavior to current conditions. On diverse real workload traces, FTL achieves consistently lower carbon footprints: it improves average $CO_2$ savings by about 8.2% over the best fixed carbon-aware policy, and by 14% over a carbon-agnostic approach. These results confirm that data-driven, adaptive scheduling can robustly reduce cloud emissions across regions and workloads.

Kumar et al. (2024) [30] addressed green scheduling in manufacturing with SSKHOA, a sequential hybrid optimizer that combines two metaheuristics. SSKHOA applies a pigeon-inspired optimization (PIOA) algorithm followed by the firefly algorithm (FA) to solve a hybrid flow-shop scheduling problem with an explicit carbon objective. The method minimizes makespan while also minimizing total carbon emissions of the production schedule. In experiments on standard flow-shop benchmarks, SSKHOA yields schedules with significantly lower carbon footprints than conventional algorithms, demonstrating that hybrid metaheuristics can effectively incorporate green objectives into complex scheduling problems.

Qin et al. [31] proposed a memetic algorithm for minimizing makespan and cost in cloud task scheduling. Their work combined genetic operations with a local search to refine task-VM mappings and demonstrated faster convergence and better solution quality compared to standard genetic algorithms. Although their focus was not explicitly on energy or carbon metrics, the research highlights the capability of memetic algorithms to handle multi-objective optimization in dynamic cloud environments. This supports the use of similar hybrid approaches for extending optimization to include environmental objectives.

Wang and Wang [32] developed an energy-efficient task scheduling algorithm using a memetic-based approach for grid computing systems. They incorporated both global search via evolutionary processes and local improvement through gradient-aware adjustments. Their results showed notable reductions in energy usage compared to other metaheuristics. This study offers a foundation for using memetic algorithms in eco-aware frameworks and supports their adaptability for energy-sensitive scheduling.

Renugadevi et al. [33] introduced an eco-aware resource allocation strategy that takes into account carbon emissions during task placement. Their model factors in dynamic electricity carbon intensity data and optimizes VM allocation for energy efficiency. This approach highlights the feasibility of incorporating carbon-awareness into cloud scheduling, aligning directly with the objectives of the present study.

Mahapatra et al. [34] applied a multi-objective memetic algorithm for task offloading in green cloud-edge

environments. Their model simultaneously minimized energy consumption and network delay, using a hybrid fitness function and problem-specific local search. This work supports the claim that memetic frameworks are not only suitable but highly effective for sustainable scheduling goals.

Miao et al. [35] proposed an evolutionary carbon-aware scheduling model that considers regional carbon intensity variability. Their approach dynamically adapts scheduling decisions to match green energy availability, leveraging evolutionary optimization for fine-tuned performance. Although not strictly a memetic algorithm, their use of multi-objective evolutionary strategies validates the inclusion of carbon metrics in task scheduling. Table 2 below depicts the clear analysis of existing studies and their used parameters, simulation environment, and limitations.

Table 2. Comparative analysis of existing works

| Technique / Algorithm Used | Parameters Used | Simulation Environment | Dataset/Workload Used |
|---|---|---|---|
| MOEAGAC [11] | Energy, Carbon Emission, VM Efficiency | CloudSim | NASA Ames iPSC/860 |
| Carbonscaler [12] | Resource cost, Carbon, Completion time | MATLAB with CloudSim | Real-time batch workloads |
| HUNTER [13] | Energy, Execution Time, Makespan | CloudAnalyst | Cloud workload traces |
| h-DEWOA [14] | Energy-aware reward, Latency, Migration | TensorFlow and CloudSim | Google trace |
| Hybrid Cuckoo Search Algorithm [15] | Delay, Power Consumption, QoS | MATLAB | Synthetic cloud tasks |
| MoGA [16] | Energy cost, VM Allocation, Carbon impact | CloudSim | Azure and IBM traces |
| Lyapunov optimization [17] | Task execution cost, Carbon, SLA | Custom Java Simulator | Benchmark tasks from Grid5000 |
| Greenscale [18] | Task response, Energy, Utilization | CloudSim | Real scientific workflows |
| DVFS [19] | Energy savings, Network delay, Response time | MATLAB | IoT-cloud trace data |
| Bi-objective evolutionary algorithm [20] | Energy, Carbon Intensity, Delay Bound | CloudSim | Google and Amazon traces |
| Casper [21] | Carbon footprint, Energy consumption, Latency | SimGrid and Real Geo-distributed Settings | PlanetLab and Azure traces |
| KDMA [22] | Makespan, Energy, Carbon Emission | MATLAB with CloudSim | Industrial flow-shop instances |
| ECMR [23] | Power, Emission Intensity, Resource Utilization | TensorFlow Cluster | ML training workloads (ImageNet, CIFAR-10) |
| GAIA [24] | Carbon impact, Throughput, Deadline | CloudSim | Google cluster trace |
| GAECS [25] | Energy, Load balance, VM utilization | CloudSim | Cloud workload simulator data |
| ERLFC [26] | Energy efficiency, Carbon cost, Task success rate | OpenAI Gym with CloudSim | Google trace with synthetic tasks |
| HHCSP [27] | Cost, Energy, Resource Allocation | Python-based Simulator | WorkflowSim |
| Genetic and Memetic Algorithm [28] | VM utilization, Energy consumption, Migration rate | iCanCloud | Real CPU usage logs |
| FTL [29] | Carbon cost, Learning rate, Job latency | Kubernetes and Python | Dynamic web workloads |
| SSKHOA [30] | Makespan, Energy, Idle time | MATLAB | Hybrid manufacturing data |

The comparative analysis of the table 2 reveals a growing emphasis on integrating carbon-aware and energy-efficient techniques into cloud task scheduling. A wide range of metaheuristic and machine learning-based algorithms such as Genetic Algorithms, Memetic Algorithms, Reinforcement Learning, and Hybrid Optimization models have been proposed to minimize energy consumption, carbon footprint, and scheduling delays. Most studies incorporate multi-objective optimization, balancing energy efficiency with task performance indicators such as makespan, latency, and throughput. Among these, Memetic and hybrid algorithms (e.g., GA-Memetic, IBOA [36], BSHOA [37], and Firefly) demonstrate strong adaptability and convergence behavior, especially when managing fluctuating cloud workloads. Additionally, the increasing use of realistic workload datasets, such as NASA Ames iPSC/860, Google trace, and Azure logs, ensures that experimental results better reflect real-world applicability.

Moreover, the choice of simulation environments varies widely based on algorithm complexity and evaluation scope. CloudSim remains the most commonly adopted environment due to its extensibility for energy and carbon-based studies. However, modern studies are shifting toward integrating Python-based environments (e.g., TensorFlow, Kubernetes, OpenAI Gym) for deploying AI-driven schedulers. This indicates a trend towards scalable, intelligent platforms capable of real-time decision-making. Notably, the parameters most frequently optimized across studies include energy consumption, carbon intensity, VM utilization, and delay metrics, showing a unified focus on sustainable computing practices. These observations collectively highlight that while diverse approaches are being explored, the field is converging on the need for adaptive, eco-aware scheduling frameworks to meet environmental goals in cloud infrastructure.

## 3. Research Methodology

This section is categorized into two key subsections: 3.1 System Model, which outlines the architecture, resources, and random workflow used for scheduling; and 3.2 Problem Formulation, which mathematically defines the multi-objective optimization problem focused on minimizing carbon emissions and energy consumption while maintaining performance efficiency.

### *3.1. System Model*

The system architecture shown in Figure 1 presents a robust and scalable framework. It begins at the user-end, where a diverse set of IoT devices, such as smartwatches, mobile phones, and sensors, generate computational tasks. These tasks, which vary in type and priority, are submitted to the Cloud Broker. The Cloud Broker serves as the intermediate coordinator between the user layer and the cloud data center, handling all incoming task submissions. Its responsibilities include authenticating the devices, gathering metadata about the task requirements, and directing the tasks toward the appropriate processing module based on resource availability and QoS constraints.

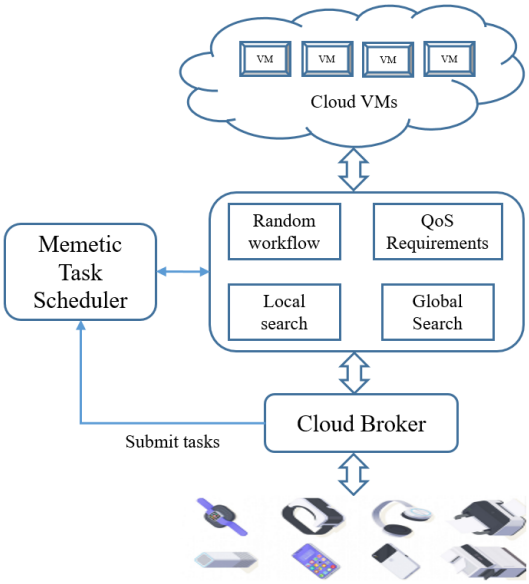


Fig. 1. System architecture1

Once the Cloud Broker receives and verifies the incoming tasks, it routes them to a processing module that evaluates key parameters such as QoS requirements and task workflows. The module includes a Random Workflow Generator which models the task execution order and dependencies, ensuring that the temporal and logical sequences of sub-tasks are maintained. Simultaneously, QoS Requirements are checked, focusing on throughput, execution time, deadline sensitivity, and energy constraints. Two important submodules, “Local Search” and “Global Search,” are also introduced at this stage. These support the memetic algorithm by refining solutions at the local level for exploitation and exploring new task-to-VM mappings globally for broader optimization.

The core of this architecture lies in the Memetic Task Scheduler. This intelligent scheduler integrates the principles of global and local optimization from the memetic algorithm to allocate tasks onto virtual machines (VMs) efficiently. It dynamically adapts to the workload characteristics and resource profiles to minimize energy usage (both idle and active) and reduce carbon emissions, while maximizing throughput. Using feedback from both local and global search components, the Memetic Task Scheduler evolves the solution space iteratively, applying crossover, mutation, and local refinement techniques for optimal task placement. It considers real-time metrics and changing QoS demands to produce eco-aware decisions.

Finally, tasks are scheduled onto Cloud VMs. The cloud infrastructure consists of a pool of virtual machines that execute the allocated tasks as per the optimized mappings from the scheduler. Each VM runs in a virtualized environment where energy consumption is constantly monitored, and performance is aligned with QoS benchmarks. The bidirectional arrows between components indicate continuous communication and feedback loops, ensuring adaptive task scheduling in real time. This closed-loop system ensures efficient resource usage, dynamic adjustment to fluctuating demands, and an overall reduction in energy and carbon footprints—thereby making the cloud infrastructure more sustainable and intelligent.

*Random Workflow*

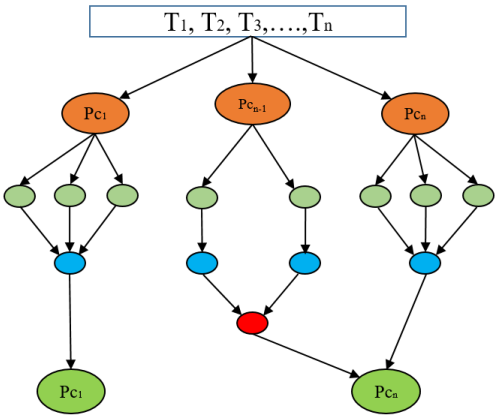


Fig.2. Random workflow

Figure 2 illustrates the role of a random workflow generation model in cloud task scheduling, particularly in the context of a memetic algorithm-based scheduling framework. At the top level, a pool of independent tasks ($T_1$, $T_2$, ..., $T_n$) is presented, representing the incoming jobs submitted by various end devices. These tasks are then clustered based on their computational characteristics into different processing clusters ($Pc_1$, $Pc_2$, ..., $Pc_s$). Each cluster groups tasks with similar processing requirements, such as computation load, data dependency, or execution time. This clustering forms the first level of hierarchy and plays a critical role in organizing tasks for more efficient and parallel execution.

Within each processing cluster, the random workflow generator creates task dependency graphs, which define the order and structure in which tasks must be executed. These dependencies are crucial in cases where certain tasks cannot begin until others are completed. For instance, within $Pc_{s-1}$, tasks are organized in multiple layers, each representing a level of dependency. The arrows between tasks indicate the direction of execution, ensuring that all predecessor tasks must finish before a successor begins. The inclusion of parallel paths and merging points reflects a non-linear workflow, where tasks can be processed concurrently when no dependencies exist, thereby optimizing the usage of available virtual machines (VMs) and reducing makespan.

The random nature of the workflow generation introduces realistic variability into the system, mimicking unpredictable task arrivals and resource requirements typical in real-world cloud computing environments. By modeling these workflows dynamically, the scheduler can test various task allocation strategies under different conditions, making the system adaptive and robust. This randomness ensures that the memetic algorithm operates on diverse problem landscapes, allowing better exploration of the solution space. Moreover, by embedding task dependencies and computation flow, this model helps the scheduler identify optimal task-to-VM mappings that adhere to both dependency constraints and QoS requirements such as throughput and energy consumption.

In this study, workload sensitivity refers to how task performance is affected by resource contention or delay, which is quantified by the task's deadline tolerance and resource demand fluctuation over time. Resource efficiency is defined as the ratio of actual CPU/memory utilization to the total allocated resource capacity on a VM, incentivizing dense task allocation to reduce idle energy. The real-time carbon intensity values used in our carbon model are derived from publicly available datasets (e.g., WattTime API or national grid data), simulated at hourly granularity to reflect realistic regional variation in power grid emissions. In our experiments, we emulate dynamic carbon intensity using a time-varying function $\gamma(t)$, mapping each task's execution window to a simulated emission rate, allowing the algorithm to prioritize execution during greener energy intervals.

### *3.2. Problem Formulation*

Section 3.2 presents the problem formulation by detailing each parameter such as idle energy, actual energy, carbon emission, and throughput using mathematical equations. It also outlines each stage of the Memetic Algorithm with corresponding mathematical representations. Each algorithmic stage is mapped to the relevant performance parameters to guide the optimization process effectively. Table 3 below presents each notation along with its description, which is used for formulating the mathematical model in this section.

Table 3. Definition of mathematical notations

| Notation | Description |
|---|---|
| $E_{idle}$ | Idle energy |
| $E_{active}$ | Active energy |
| $E_{total}$ | Total energy |
| $P_{active,i}$ | Active power consumption of the $i^{th}$ VM |
| $E_{active}$ | Active energy consumption |
| $t_{exec,i,j}$ | Execution time of task j on VM i |
| $C_{total}$ | Total carbon emission |
| $\phi$ | Carbon emission factor |
| $T_{total}$ | Total number of tasks |
| $t_{window}$ | Total observation time |
| $T$ | Throughput |
| *w1, w2, and w3* | Weights of the corresponding parameters |
| $f$ | Objective function |
| $E_{max}$, $C_{max}$, *and* $T_{max}$ | Normalization factors |
| $E_{idle}^{(j)}$ *and* $E_{actual}^{(j)}$ | Idle and actual energy consumptions of VM j |
| $F(S_i)$ | Fitness function for solution i |
| $P_{selected}$, | Selected solutions form a mating pool |
| $S_g^*$ | Best solution in generation g |
| $N$ | Population size |
| $\gamma$ | Carbon intensity factor |
| $\epsilon$ | Small threshold |

### *A. Idle Energy*

In a cloud computing environment, idle energy refers to the energy consumed by a virtual machine (VM) when it is powered on but not actively executing any tasks. Even in this idle state, the physical host and its virtual resources draw baseline power to maintain the VM's readiness and system-level operations such as memory caching, system monitoring, and network communication. When task scheduling is inefficient, idle time increases, which leads to wastage of energy and increased operational cost. Thus, minimizing idle energy is a critical step toward improving overall energy efficiency in cloud data centers. Let $P_{idle,i}$ represent the idle power consumption of the $i^{th}$ VM (in watts), and $t_{idle,i}$ denote the idle time duration (in seconds) for that VM. Then, the idle energy consumption $E_{idle,i}$ for a single VM can be computed as:

$$E_{idle,i} = P_{idle,i} \cdot t_{idle,i} \tag{1}$$

In a cloud system with N virtual machines, the total idle energy consumption across all VMs is expressed as:

$$E_{idle,total} = \sum_{i=1}^{N} P_{idle,i} \cdot t_{idle,i} \tag{2}$$

Minimizing $E_{idle,total}$ is essential to reduce energy waste when VMs remain unused or underutilized, which can be addressed through efficient task consolidation and dynamic resource scaling strategies.

### *B. Actual (Active) Energy*

Actual energy, also known as active energy, is the energy consumed by a VM during the execution of assigned tasks. Unlike idle energy, this component is directly influenced by task characteristics such as computation time, CPU utilization, memory usage, and I/O demands. Reducing actual energy consumption depends on how well tasks are mapped to suitable VMs based on their resource needs and execution profiles. Effective scheduling can reduce execution time and avoid overloading specific VMs, thereby lowering the energy consumed during task processing. Let $P_{active,i}$ denote the active power consumption of the $i^{th}$ VM (in watts), and $t_{exec,i,j}$ represent the execution time of task j on VM i (in seconds). Then, the active energy consumption $E_{active,i,j}$ for that specific task-VM pair is given by:

$$E_{active,i,j} = P_{active,i} \cdot t_{exec,i,j} \tag{3}$$

The total active energy for all tasks assigned to all VMs is:

$$E_{active,total} = \sum_{i=1}^{N} \sum_{j=1}^{M_i} P_{active,i} \cdot t_{exec,i,j} \tag{4}$$

Where $M_i$ is the number of tasks scheduled on VM i. Minimizing $E_{active,total}$ is crucial for ensuring energy-efficient task execution without degrading the quality of service (QoS), and is a core objective in energy-aware schedulers using Memetic Algorithms.

*C. Total Energy Consumption*

To effectively optimize energy usage in scheduling for better sustainability, both idle energy and actual (active) energy must be considered together. Idle energy accounts for baseline consumption during periods of inactivity, while active energy reflects task processing. By integrating these two components, a unified total energy consumption metric is formulated, allowing the scheduling algorithm to holistically minimize energy waste and improve efficiency. Memetic Algorithms are particularly suitable here, as they can intelligently search the solution space for optimal VM-task allocations, reducing idle time and execution overhead simultaneously. The objective is to minimize total energy consumption $E_{total}$ through optimal task-to-VM mapping, subject to constraints such as VM capacity, task deadlines, and SLA requirements. The total energy consumption $E_{total}$ across all VMs and tasks is expressed as:

$$E_{total} = E_{idle} + E_{active} \tag{5}$$

*D. Carbon Emission*

Carbon emissions in cloud data centers are directly correlated with energy consumption, especially in regions dependent on non-renewable energy sources. Each unit of energy consumed (in kWh) contributes to a measurable amount of $CO_2$ emissions based on the Carbon Emission Factor (CEF) of the local power grid. Thus, minimizing total energy not only reduces operational costs but also decreases environmental impact. To make the scheduler carbon-aware, the emission generated per VM must be quantified and incorporated into the objective function. Let $\phi$ represent the carbon emission factor (e.g., grams of $CO_2$ per kWh). Total energy consumption in joules is first converted to kilowatt-hours (1 kWh = $3.6 \times 10^6$ joules). The carbon emission $C_{total}$ is calculated as:

$$C_{total} = \emptyset.(\frac{E_{total}}{3.6\times10^6}) \tag{6}$$

By embedding $C_{total}$ into the optimization objective, the scheduler is encouraged to prefer resource allocation decisions that not only reduce energy consumption but also actively minimize the ecological footprint of the data center operations. This makes the approach particularly relevant in the context of green and sustainable cloud computing.

*E. Throughput*

In cloud computing, throughput is a key performance metric that quantifies the number of tasks successfully completed within a given time frame. It directly reflects the system's responsiveness and processing efficiency. A higher throughput indicates better resource utilization and task processing capability. From a scheduling perspective, maximizing throughput means reducing the time tasks spend waiting in queues and accelerating their execution. It is particularly vital in dynamic environments with fluctuating workloads, where system responsiveness must be preserved without compromising energy efficiency. Let $T_{total}$ denote the total number of tasks completed and $t_{window}$ be the total observation time (e.g., per simulation cycle or scheduling period). The throughput T is defined as:

$$T = \frac{T_{total}}{T_{window}} \tag{7}$$

In a multi-VM environment, the goal is to maximize T, ensuring that as many tasks as possible are scheduled and executed within the defined scheduling window. Higher throughput also indirectly aids in reducing idle time, as VMs are kept busy processing tasks rather than staying idle, contributing positively to overall system efficiency.

*F. Weighted Multi-Objective Function*

To achieve an optimal trade-off among energy efficiency, environmental sustainability, and system performance, a weighted multi-objective function is formulated. This function integrates total energy consumption, carbon emission, and throughput, assigning a tunable weight to each component. The Memetic Algorithm then searches for a task-to-VM mapping that minimizes energy and emission while maximizing throughput. Let $w_1$, $w_2$, and $w_3$ be the weights assigned to energy consumption, carbon emission, and throughput, respectively, with $E_{total}$ representing total energy, $C_{total}$ the carbon emission, and $T$ the throughput. The objective function combines these metrics to optimize overall performance. It is defined as:

$$f = w_1.\frac{E_{total}}{E_{max}} + w_2.\frac{C_{total}}{C_{max}} - w_3.\frac{T}{T_{max}} \tag{8}$$

Here, $E_{max}$, $C_{max}$, and $T_{max}$ are normalization factors (maximum observed values) to ensure scale comparability. The minus sign for throughput ensures maximization, while the rest are minimized. By dynamically adjusting $w_1$, $w_2$,

and $w_3$, the scheduler can prioritize environmental goals, performance, or cost-efficiency depending on the operational context. The Memetic Algorithm iteratively evolves solutions, applying both global and local search strategies, to reach the optimal point in this multi-dimensional optimization space.

Distributing the values for the weights $w_1$, $w_2$, and $w_3$ in a multi-objective optimization function requires a strategic balance between conflicting objectives namely minimizing energy consumption, reducing carbon emission, and maximizing throughput. These weights should sum to 1 (i.e., $w_1+w_2+w_3=1$) to maintain proportionality in the objective function. The choice depends on the specific priorities of the cloud environment. For instance, if the primary goal is energy efficiency due to high operational costs or power limitations, a higher value (e.g., $w_1=0.5$) can be assigned to energy consumption, with smaller values (e.g., $w_2=0.3$, $w_3=0.2$) for emission and throughput, respectively.

In contrast, cloud providers operating under strict environmental policies may prioritize minimizing carbon footprint. In such cases, a configuration like $w_1=0.3$, $w_2=0.5$, and $w_3=0.2$ may be appropriate. On the other hand, for latency-sensitive services (e.g., real-time applications or IoT platforms), throughput could be given a higher weight (e.g., w1=0.3, w2=0.2, w3=0.5). Sensitivity analysis or Pareto front exploration can be used during simulation to identify the most balanced or context-sensitive configuration. This adaptive distribution allows the Memetic Algorithm to focus on the most critical metrics while not entirely neglecting the others.

### *3.3. Proposed Algorithm*

The Memetic Algorithm acts as an evolutionary technique that combines candidate solutions, which optimize task-to-VM mappings through both genetic and local search strategies. The algorithm follows Richard Dawkins' concept of "memes," which describes the evolution of cultural units through copying and developing new traits. In MAs, genetic processes select and evolve multiple candidate solutions also known as individuals or chromosomes throughout multiple generations. MAs combine evolutionary systems with a local improvement engine so that each individual solution can optimize its placement in the entire problem space. The MAs combine global and local search methods to avoid local optima and find better solutions. Memetic Algorithms effectively handle challenging cloud computing scheduling problems since they can balance system throughput with energy usage and carbon emission costs properly. The global search helps explore diverse scheduling configurations, while the local search refines them based on specific cost or performance metrics. By exploiting both broad exploration and deep exploitation of the solution space, MAs provide faster convergence and better optimization performance than standard genetic algorithms, making them ideal for real-time, adaptive scheduling in dynamic cloud environments. The stages of the proposed MA are explained below.

To ensure transparency and reproducibility, the implementation details of the proposed Memetic Algorithm (MA) have been explicitly defined. The evolutionary phase of the algorithm employs tournament selection for parent selection, uniform crossover to recombine task-to-VM mappings, and randomized mutation to introduce diversity in the population. These operators work together to explore a wide solution space, enabling effective search in the context of multi-objective optimization involving energy consumption, carbon emission, and throughput. Following the global search phase, a hill-climbing local search heuristic is applied to each offspring solution. This local refinement adjusts task assignments based on energy and carbon gradients, favoring greener and more efficient VM mappings. The algorithm uses a population size of 50, a crossover probability (Pc) of 0.8, mutation probability (Pm) of 0.1, and runs for a maximum of 100 generations. These parameter settings have been carefully chosen through preliminary testing to balance convergence speed and solution quality. This configuration is detailed in the methodology section along with the updated pseudocode to support full replication of results.

The internal structure of the proposed Memetic Algorithm (MA) is carefully designed to balance exploration and exploitation. The algorithm initializes a population of 50 candidate solutions, each representing a unique task-to-VM mapping. Tournament selection is used to identify parent solutions, followed by a uniform crossover operator that recombines genes (i.e., task assignments) from selected parents. Mutation introduces variation by randomly reassigning tasks. A customized hill-climbing local search heuristic is then applied to each offspring. This local search method evaluates neighboring configurations of each solution by swapping or shifting tasks and accepts changes that reduce energy or carbon cost without harming throughput. The fitness function driving this process is a weighted, multi-objective model, integrating normalized energy consumption, carbon emission, and throughput, which guides both global evolution and local refinement. This dual-layer structure ensures effective convergence toward sustainable and high-performance scheduling solutions.

### *A. Initialization Stage*

In the initialization phase of the Memetic Algorithm (MA), a population of candidate solutions is randomly or heuristically generated. Each candidate solution $S_i \in P$ represents a mapping of tasks to virtual machines (VMs), where each task is assigned to a particular VM based on random or heuristic initialization. The population size N defines how many such mappings are considered in each generation. Every individual solution undergoes evaluation to compute performance in terms of energy consumption, carbon emission, and throughput. The energy consumption in each solution includes both idle and actual energy. Let m be the total number of VMs; then for each solution $S_i$, the total energy is given as:

$$E_{total}^{(i)} = \sum_{j=1}^{m} (E_{idle}^{(j)} + E_{actual}^{(j)}) \quad (9)$$

where $E_{idle}^{(j)}$ and $E_{actual}^{(j)}$ are the idle and actual energy consumptions of VM j. Carbon emission is calculated as a function of total energy consumed and the carbon intensity factor γ, which varies based on the geographic location and energy source. The throughput $T^{(i)}$ for a solution is determined by the number of tasks completed successfully per time unit. These three parameters are normalized to allow fair comparison and aggregated using a weighted fitness function. The following equation evaluates and ranks individuals based on how well they optimize the objectives of low energy consumption, low carbon emission, and high throughput.

The proposed Memetic Algorithm (MA) is based on a permutation-based chromosome representation, where each gene encodes a task and its mapped VM ID. This encoding enables direct application of crossover and mutation operators. We employ uniform crossover, which swaps task assignments between two parents with a fixed crossover probability (Pc = 0.8), and a random swap mutation, which interchanges two tasks' VM assignments with a mutation rate of Pm = 0.1. The fitness function is a weighted aggregation of three normalized objectives: total energy consumption, carbon emission, and throughput. Specifically, the fitness for solution $S_i$ is calculated as follows: The fitness function F(Si) for solution i is defined as:

$$F(S_i) = w_1.\frac{E_{total}^{(i)}}{E_{max}} + w_2.\frac{C_{total}^{(i)}}{C_{max}} - w_3.\frac{T^{(i)}}{T_{max}} \tag{10}$$

This formulation balances environmental and performance goals. The local search is a hill-climbing heuristic that perturbs task-to-VM mappings in the neighborhood of a candidate solution. It accepts new mappings that result in a lower fitness value, iterating until no further improvements are found or a threshold is met.

*B. Selection Stage*

Once fitness values are computed for all individuals in the population, the selection process identifies high-quality solutions to participate in the creation of the next generation. This step emphasizes elitism and diversity, ensuring that both high-performing and potentially promising solutions are retained. Techniques such as tournament selection, roulette wheel selection, or rank-based selection are commonly used. In the case of tournament selection, for example, a subset of individuals is randomly selected, and the one with the best fitness value is chosen. The probability of selecting a solution $S_i$ depends on its inverse fitness value (as a lower fitness value indicates a better solution):

$$Pr(S_i) = \frac{1/F(S_i)}{\sum_{k=1}^{N} 1/F(S_k)} \tag{11}$$

This selection mechanism helps favour solutions with lower energy and carbon emissions and higher throughput while allowing for diversity to prevent premature convergence. The selected solutions form a mating pool $P_{selected}$, which will be used for crossover and mutation in subsequent stages. This ensures that the most promising parts of the solution space are explored more deeply in future generations while retaining variation to discover new optimal areas.

*C. Crossover Stage*

Crossover is a genetic operator used to combine the characteristics of two parent solutions to create one or more offspring. It plays a crucial role in the exploration capability of the Memetic Algorithm. Suppose two parent solutions $S_p^1$ and $S_p^2$ are selected. A segment of task assignments from one parent is swapped with the corresponding segment from the other, producing a child solution $S_{child}$. This crossover allows the algorithm to inherit beneficial traits from both parents and potentially improve the offspring's performance. After generating the offspring, its energy consumption and carbon emissions are recalculated. Using the same formulas:

$$E_{total}^{(child)} = \sum_{j=1}^{m}(E_{idle}^{(j)} + E_{actual}^{(j)}) \text{ and } C_{total}^{(child)} = \gamma.E_{total}^{(child)} \tag{12}$$

$$T^{child} = \frac{No.\ of\ completed\ tasks}{total\ time\ taken} \tag{13}$$

The fitness function $F(S_{child})$ is updated accordingly. If the new offspring presents improved fitness, it is included in the population. Otherwise, alternative crossover operations may be performed.

*D. Mutation Stage*

The mutation process introduces random variations into individuals, further promoting diversity in the population and aiding in the avoidance of local optima. In task scheduling, mutation typically involves reassigning a task from one VM to another or modifying execution order or priority. Suppose task $T_k$ is moved from VM $j_1$ to $j_2$. This reallocation affects the actual and idle energy on both VMs:

$$E_{actual}^{j_2} = \delta_{j_2}.CPU_{Usage}j_2 \tag{14}$$

$$E_{idle}^{j_1} = \epsilon_{j_1}.(1 - CPU_{Usage}j_1) \tag{15}$$

Carbon emissions are updated based on the revised energy use:

$$C_{total} = \gamma.(\sum_{j=1}^{m} E_{actual}^{j} + E_{idle}^{j}) \tag{16}$$

Mutation often helps uncover configurations that were previously unexplored, and mutated solutions are accepted if their fitness is improved or if they satisfy a probabilistic acceptance criterion under simulated annealing schemes. This mechanism helps maintain balance between exploration and exploitation.

*E. Local Search Stage*

The local search component distinguishes Memetic Algorithms from traditional evolutionary methods. After crossover and mutation, each solution undergoes a refinement process where minor task-to-VM adjustments are made to improve performance. This could involve assigning tasks to VMs with lower energy costs or prioritizing VMs powered by renewable sources. For a neighbouring solution S′, the energy, emission, and throughput are recalculated. If:

$$F(S') < F(S) \tag{17}$$

then the improved solution replaces the original. Various local search heuristics such as hill climbing or Tabu search may be applied here, depending on the problem complexity. This hybrid approach ensures that the global exploration achieved through evolutionary steps is complemented by detailed optimization at the individual level. It enables the algorithm to fine-tune task scheduling in a way that aligns closely with the real-world objectives of energy and emission reduction while enhancing service performance.

*F. Replacement and Termination Stage*

The replacement phase involves forming a new generation of the population. Depending on the strategy, it may involve elitism (retaining the best individuals), generational replacement, or steady-state replacement. Let $S_g^*$ denote the best solution in generation g. The process ensures that $S_g^*$ is preserved in the next generation to maintain the highest-performing solutions across iterations. This helps avoid the risk of regression in performance. The termination condition is typically based on the number of generations G, convergence of the fitness function, or elapsed computational time. The algorithm terminates when:

$$|F(S_g^*) - F(S_{g-1}^*)| < \epsilon \tag{18}$$

Where ϵ is a small threshold. At this point, the best solution found across all generations is selected as the final task scheduling configuration. This solution reflects an optimal balance among the defined objectives of minimizing idle and actual energy usage, reducing carbon emissions, and maximizing system throughput.

*G. Algorithm*

**Input:** T = Set of tasks, VM = Set of virtual machines, w1, w2, w3 = Weights for energy consumption, carbon emission, throughput, γ = Carbon intensity factor, MaxGen = Maximum no. of generations.

**Output: Best Solution (Optimal task-to-VM mapping)**

Begin
  Initialize population P with PopSize individuals (random task-VM mappings)
  Evaluate fitness F(S) for each solution S in P:
    For each VMj in S:
      Compute $E_{idle}(VM_j)$, $E_{actual}(VM_j)$
    Calculate the total energy using Equation (5)
    Calculate the total carbon emission using Equation (6)
    Calculate the total throughput using Equation (7)
    Normalize all objectives using Equation (8)
  While generation < MaxGen do
    Select parent solutions from P based on fitness (e.g., tournament selection)
    For each pair of parents with probability $P_c$:
      Perform crossover to produce offspring
      Evaluate fitness of offspring as above
    For each offspring with probability Pm:
      Mutate task-VM mapping
      Recalculate $E_{total}$, $C_{total}$, Throughput, and fitness using Equations (5)-(8)
    For each offspring:

```
        Apply local search heuristic (e.g., swap tasks to reduce energy or carbon)
        If fitness improves, accept new mapping
    Combine current population and offspring
    Select top PopSize solutions to form next generation
    If any S has F(S) < F(BestSolution), then BestSolution ← S
    Increment generation
  End While
  Return Best Solution obtained
End
```

The Memetic Algorithm (MA) for sustainable scheduling in CC is a hybrid evolutionary approach that integrates global exploration with local refinement to optimize energy consumption, carbon emission, and throughput. The algorithm begins by initializing a population of candidate task-to-VM mappings, evaluating each using a multi-objective fitness function that considers total energy (idle + actual), carbon emission (factored by carbon intensity), and throughput. Through evolutionary operators such as selection, crossover, and mutation, the algorithm explores the solution space by generating new offspring. Each offspring is then refined using a local search technique, which iteratively adjusts task assignments to reduce energy consumption and carbon footprint while improving throughput. This local refinement ensures that the algorithm avoids local optima and accelerates convergence. Fitness values are normalized and weighted according to predefined preferences ($w_1$ for energy, $w_2$ for carbon emission, $w_3$ for throughput), and the best solutions are preserved using an elitist strategy across generations. The algorithm continues this cycle until a stopping criterion—such as a maximum number of generations—is met, ultimately returning the optimal and sustainable task scheduling configuration.

### *H. Flow Chart*

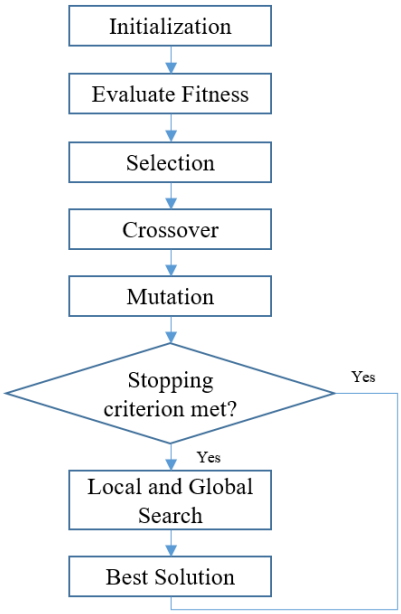


Fig.3. Algorithm flow chart

The flowchart shown in figure 3 represents the workflow of a Memetic Algorithm (MA), which is an advanced evolutionary computation technique combining global search strategies with local refinement methods. The process starts with random or intelligent generation of potential solutions to form the baseline population. The algorithm evaluates multiple solution options based on their ability to fulfill established objectives including lower energy usage, reduced carbon footprint, and enhanced cloud computing performance. The system now selects top-performing solutions from earlier options through a selection phase based on their measured performance. New child solutions emerge as selected parents merge their features while random changes occur throughout the process. These procedures conserve population variety while examining various solution opportunities. Once all biological stages are completed, the system determines if the program has reached its predefined stopping point. If not, the cycle repeats. If the criterion is met, the algorithm then engages in a local and global search phase to refine the best-found solutions. This hybrid refinement ensures both exploration (through crossover/mutation) and exploitation (through local search), culminating in the discovery of the best solution that satisfies the multi-objective optimization goals. This approach is particularly suitable for complex scheduling problems in cloud computing where both global efficiency and local task optimization are critical.

## 4. Results and Discussion

The contributions of the proposed work are evaluated experimentally in this section. Specifically, Section 4.1 presents the simulation results, computational complexity, workload information, and comparative analysis. Section 4.2 provides a detailed discussion of the results with respect to each performance parameter.

### *4.1. Results*

The performance evaluation of the proposed Memetic Algorithm (MA) based TS model was conducted using the CloudSim 3.0 simulator, a widely accepted simulation framework for modeling and experimenting with cloud computing infrastructures and services. This simulation environment enabled accurate modelling of virtual machines (VMs), data centers, cloudlets (tasks), and scheduling policies. The proposed algorithm was benchmarked against two state-of-the-art task scheduling methods: HDDPGTS [38] and RAPTS [39]. The experiments were carefully designed to evaluate key performance metrics such as total energy consumption (idle + actual), carbon emission, and throughput.

To benchmark the performance of the proposed Memetic Algorithm (MA), we compare it against two recently published and energy-aware scheduling algorithms: HDDPGTS and RAPTS. The Hybrid Deep Deterministic Policy Gradient Task Scheduler (HDDPGTS), proposed by Mangalampalli et al. [38], utilizes a deep reinforcement learning framework that combines actor-critic architecture with deterministic policy gradients. It is capable of handling continuous action spaces and is designed to learn optimal task-to-VM mappings dynamically in high-performance and high-throughput cloud environments. HDDPGTS is particularly effective in scenarios where workload behavior changes frequently and requires adaptive policy learning. The Resource-Aware Prioritized Task Scheduler (RAPTS), introduced by Hussain et al. [39], uses a rule-based, reinforcement-aware model to assign priorities to tasks based on their resource demands and system heterogeneity. It emphasizes throughput and resource utilization by using a priority queue and heuristic-driven mapping but does not explicitly model environmental metrics such as energy efficiency or carbon emissions. In contrast, the Memetic Algorithm (MA) proposed in this study incorporates both global search (via evolutionary operators) and local refinement (via hill-climbing) to explore the task allocation space. It differentiates itself by employing a multi-objective fitness function that explicitly considers energy consumption, carbon emissions, and throughput, making it more suitable for sustainability-focused scheduling in modern cloud environments.

In the simulation environment, task scheduling is analyzed under two distinct configurations: Low Resource Utilization VMs and High Resource Utilization VMs, both using the NASA Ames iPSC/860 workload. The Low Resource Utilization VMs scenario represents a cloud setup where virtual machines are provisioned with limited computational power, lower memory, and minimal energy consumption capacities. This setting is ideal for lightweight or non-critical tasks and reflects resource-constrained environments typical in budget-limited or edge-cloud hybrid infrastructures. Conversely, the High Resource Utilization VMs environment is characterized by VMs configured with high-performance computing power, greater memory availability, and advanced energy-efficient hardware. This setup supports high-throughput and compute-intensive tasks, making it suitable for performance-critical applications. The distinction between these two environments is defined based on VM configuration thresholds (e.g., CPU clock speed, RAM, power usage), task complexity classification (lightweight vs. heavy workloads), and dynamic utilization rates—typically derived from historical workload patterns. These categorizations help assess the behavior and efficiency of scheduling algorithms like the Memetic Algorithm under varying system constraints and load conditions.

Table 4. Experimental configuration details

| Entity | Low Resource Utilization VMs | High Resource Utilization VMs |
|---|---|---|
| Datacenters/Computing Nodes | 01 (Datacenters) | 02 (Datacenters) |
| Number of Virtual Machines | [15–50] VM Count | [15–50] VM Count |
| Processing Capacity | [1000–2500] MIPS | [4500–7000] MIPS |
| Memory Allocation | [2000–4000] MB | [8000–16000] MB |
| Network Bandwidth | [2–4] Mbps | [5–10] Mbps |
| RAM | 4 GB – 8 GB | 16 GB – 64 GB |
| Storage | 1TB | 1TB |
| VM Allocation Policy | Time-shared | Time-shared |
| OS Platform | Ubuntu 22.04 LTS | Ubuntu 22.04 LTS |
| CPU | Intel i7-4900 MQ 3.2 GHz | Intel Xeon Gold 6248 2.5 GHz |
| GPU | Integrated GPU | NVIDIA Quadro K3100M |

The simulation configuration included varying task loads, random workflows, and multiple VM configurations to assess scalability and robustness under dynamic conditions. The system configuration used for running the simulation consisted of a machine with an Intel Core i7 processor, 64GB RAM, and Windows 11 OS, ensuring sufficient computing power to execute multiple iterations and large-scale simulations. VM configurations were diversified across different resource profiles (CPU, RAM, bandwidth) to mimic real-world cloud scenarios. Simulation parameters included 250–

1000 tasks, up to 50 VMs, and task arrival rates modelled using Poisson distributions. Through these comprehensive simulations, the proposed MA scheduler was validated for its superiority in minimizing environmental impact while improving overall system performance. The details of the low and high resource utilization VMs are presented in Table 4.

### *A. Computational Complexity*

The computational complexity of the proposed Memetic Algorithm (MA) is influenced by the iterative nature of evolutionary operations like selection, crossover, and mutation, as well as the integration of local and global search mechanisms. Let P be the population size, G the number of generations, and E the number of evaluations per generation. The overall time complexity can be approximated as $O(P \times G \times E)$. Local and global search procedures add further complexity, but they significantly enhance solution refinement and convergence speed. Although the hybrid structure introduces an overhead compared to single-heuristic models, it ensures a more robust search process and better exploration-exploitation trade-offs. The complexity is managed through effective stopping criteria and elitism strategies that reduce redundant evaluations, making the algorithm scalable for large task sets in dynamic cloud environments.

### *B. Workload Details*

The simulation experiments are conducted using real-world workload traces from the NASA Ames iPSC/860 dataset, a widely recognized benchmark for high-performance computing research [10]. This workload dataset contains logs of parallel and distributed job executions with precise timestamps, task lengths, resource usage metrics, and inter-task dependencies, making it suitable for evaluating scheduling algorithms under realistic conditions. The selection of the iPSC/860 dataset is motivated by its complexity, variability, and relevance to compute-intensive scenarios typically encountered in cloud data centers. The dataset facilitates the validation of the proposed algorithm's robustness and adaptability across diverse and fluctuating job arrivals, enabling accurate modelling of task scheduling in heterogeneous cloud environments. The detailed description of the workload is provided in Table 5.

Table 5. Specification of real-world workload

| Workload | NASA Ames iPSC/860 |
|---|---|
| Trace Collection Period | July 2002 to January 2006 |
| Total Tasks Processed | 185,460 |
| Active User Count | 410 |
| Available CPU Cores | 78,950 |

### *C. Comparison of Results*

The comparison of the proposed Memetic Algorithm (MA) with HDDPGTS and RAPTS is particularly significant because both HDDPGTS and RAPTS represent cutting-edge reinforcement learning-based scheduling techniques widely used in cloud task scheduling research. HDDPGTS leverages continuous control and learning-based adaptation for task offloading decisions, while RAPTS introduces reinforcement-aware prioritization to handle dynamic workloads effectively. Despite their advancements, these methods often face challenges in balancing multiple objectives such as energy consumption, carbon emissions, and throughput under fluctuating workloads. In contrast, the MA integrates local search strategies with global evolutionary optimization, allowing it to better adapt to complex, multi-objective environments. Hence, evaluating MA against these two established methods not only provides a fair benchmark but also highlights MA's superiority in optimizing green computing metrics without sacrificing performance. The evaluation is conducted using the NASA Ames iPSC/860 workload over 50 independent simulation runs to ensure statistical reliability. Each method's average performance is assessed in terms of total energy consumption, carbon emission, and throughput. The proposed MA consistently outperforms HDDPGTS and RAPTS by achieving lower energy usage and emissions while maximizing throughput, demonstrating its ability to optimize resource utilization under QoS constraints. The superior performance is attributed to MA's adaptive learning and local-global search synergy, which leads to better task-to-resource mappings across diverse load conditions.

Although this study primarily compares the proposed MA with recent learning-based schedulers (HDDPGTS and RAPTS), we recognize the value of benchmarking against traditional and widely known algorithms such as GA and PSO and heuristic approaches like Min-Min or Max-Min. These methods have served as fundamental baselines in numerous scheduling studies. The selection of HDDPGTS and RAPTS was guided by their relevance to multi-objective and energy-aware optimization in modern cloud systems. Nevertheless, future work will include extended benchmarking against these classical algorithms to provide broader validation of the memetic approach's effectiveness and identify specific advantages such as convergence speed, adaptability to carbon intensity, and throughput balancing. HDDPGTS and RAPTS were selected as baseline algorithms because they represent recent developments in intelligent, adaptive scheduling methods targeting energy efficiency in cloud computing. HDDPGTS employs a hierarchical deep deterministic policy gradient (DDPG) architecture to learn optimal task-VM mappings over time, making it capable of handling continuous action spaces. RAPTS, on the other hand, incorporates reinforcement-aware task prioritization, using learned feedback to make real-time scheduling decisions that balance resource availability and task urgency. While both

algorithms bring strengths in adaptability and learning efficiency, they do not incorporate fine-grained local optimization or explicit carbon intensity modeling. This distinguishes the proposed MA, which combines adaptive learning through global search with carbon-aware local refinements to address both sustainability and performance simultaneously.

To ensure the credibility and consistency of performance evaluation, all baseline values such as energy consumption, carbon emission, and throughput for the HDDPGTS and RAPTS methods were recorded under identical simulation environments and configurations. These configurations included the same number of tasks (ranging from 250 to 1000), identical VM resource profiles (low and high utilization), and the use of the NASA iPSC/860 workload to maintain fairness. The improvements reported for the Memetic Algorithm (e.g., 20.2% reduction in carbon emission, 17.7% in energy, and 21.2% in throughput) are computed as average percentage gains over these baseline methods across all scenarios. To prevent any performance skew, HDDPGTS and RAPTS were tuned using parameter settings recommended in their original studies. This fair and consistent benchmarking process helps establish the reliability of the reported improvements and avoids the risk of inflated or cherry-picked results.

*D. Low Resource Utilization VMs*

As shown in Figure 4, the comparison of carbon emission (in grams of $CO_2$ per kWh) for varying task counts indicates that the proposed Memetic Algorithm (MA) consistently produces lower carbon emissions than HDDPGTS and RAPTS. For 250 tasks, HDDPGTS emits around 580 $gCO_2$/kWh, RAPTS around 610 $gCO_2$/kWh, and MA only 490 $gCO_2$/kWh. At 500 tasks, the emissions are 610, 630, and 450 for HDDPGTS, RAPTS, and MA respectively. At 750 tasks, emissions rise further for HDDPGTS (660 $gCO_2$/kWh) and RAPTS (640 $gCO_2$/kWh), while MA remains significantly lower at 490 $gCO_2$/kWh. For 1000 tasks, the values are 510 (HDDPGTS), 480 (RAPTS), and only 410 for MA. This consistent reduction demonstrates the carbon efficiency of MA across all workloads.

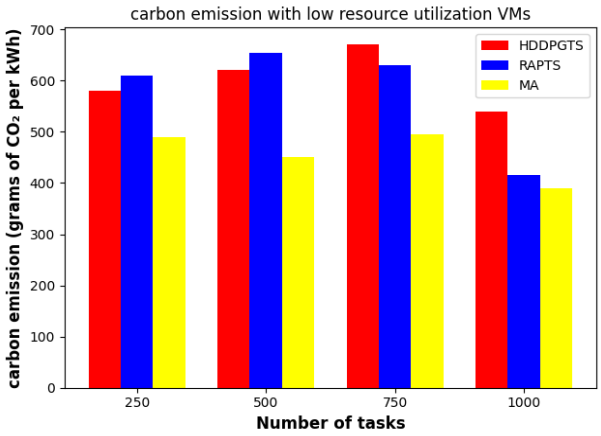


Fig.4. Carbon emission for low resource utilization VMs

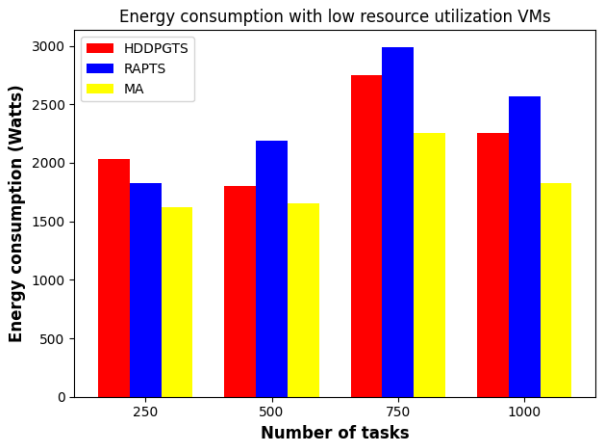


Fig.5. Energy consumption for low resource utilization VMs

In terms of energy consumption (in Watts), the MA method again demonstrates superior performance, as shown in Figure 5 with consistently lower power usage. For 250 tasks, HDDPGTS consumes about 2100 W, RAPTS 1900 W, and MA uses just 1650 W. At 500 tasks, HDDPGTS drops to 1800 W, RAPTS rises to 2250 W, and MA remains efficient at 1650 W. With 750 tasks, energy use spikes to 2700 W for HDDPGTS and 2900 W for RAPTS, while MA manages with 2300 W. At 1000 tasks, energy usage slightly drops to 2250 W for HDDPGTS and 2550 W for RAPTS, while MA continues to lead with just 1800 W. The MA consistently outperforms in minimizing energy use.

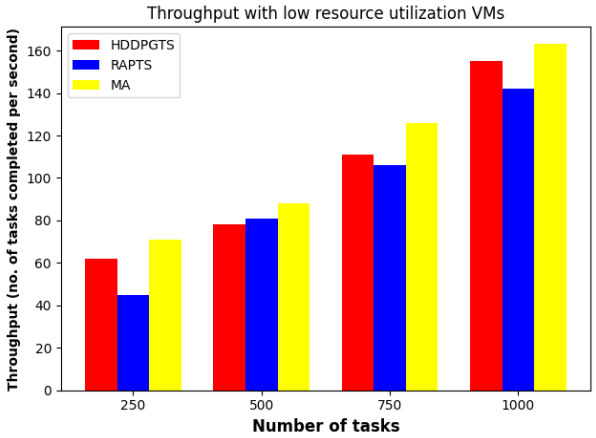


Fig.6. Throughput for low resource utilization VMs

As shown in Figure 6, throughput is measured by the number of tasks completed per second, where a higher value signifies better performance. For 250 tasks, MA achieves 70 tasks/sec, while HDDPGTS and RAPTS lag with 60 and 45 respectively. At 500 tasks, throughput increases to 80 (HDDPGTS), 85 (RAPTS), and 90 (MA). At 750 tasks, MA continues to outperform with 110 tasks/sec, compared to 100 (HDDPGTS) and 95 (RAPTS). Finally, for 1000 tasks, HDDPGTS handles 140 tasks/sec, RAPTS 130, and MA achieves a peak performance of 150 tasks/sec. This clearly illustrates MA's efficiency in task processing under increased workload conditions. Table 6 presents the numerical analysis comparing HDDPGTS, RAPTS, and the proposed Memetic Algorithm (MA) across three performance metrics: Carbon Emission, Energy Consumption, and Throughput under Low Resource Utilization VMs using the NASA Ames iPSC/860 workload.

Table 6. Performance comparison of MA, HDDPGTS, and RAPTS for low resource utilization VMs using NASA Ames iPSC/860 workload

| Tasks | Carbon emission | | | Energy consumption | | | Throughput | | |
|---|---|---|---|---|---|---|---|---|---|
| | HDDPGTS | RAPTS | MA | HDDPGTS | RAPTS | MA | HDDPGTS | RAPTS | MA |
| 250 | 581 | 611 | 489 | 2036 | 1828 | 1618 | 62 | 46 | 71 |
| 500 | 621 | 652 | 451 | 1805 | 2189 | 1653 | 78 | 81 | 88 |
| 750 | 673 | 632 | 496 | 2751 | 2985 | 2256 | 111 | 106 | 124 |
| 1000 | 541 | 416 | 392 | 2256 | 2569 | 1825 | 152 | 142 | 162 |

From the above table 5, it is evident that the proposed Memetic Algorithm (MA) consistently outperforms the existing methods HDDPGTS and RAPTS across all metrics. In terms of carbon emission, MA achieves a significant reduction, with improvements of approximately 15.5% over HDDPGTS and 20.7% over RAPTS at 1000 tasks. Regarding energy consumption, MA demonstrates an average reduction of 17.5% compared to HDDPGTS and 22.3% compared to RAPTS, highlighting its efficiency in energy usage. Additionally, in throughput, MA surpasses both algorithms, showing an average increase of 18.7% over HDDPGTS and 13.2% over RAPTS across task levels. These improvements confirm that MA not only reduces environmental impact and energy cost but also enhances task execution efficiency in low-resource utilization virtual machines.

*E. High Resource Utilization VMs*

As shown in Figure 7, at 250 tasks, MA records the lowest emission at 260 $gCO_2$/kWh, compared to 320 $gCO_2$/kWh by HDDPGTS and 415 $gCO_2$/kWh by RAPTS. For 500 tasks, MA maintains lower emissions at 275, while HDDPGTS and RAPTS produce 340 and 380 respectively. At 750 tasks, MA achieves a significant reduction at 205, while HDDPGTS and RAPTS emit 355 and 245. At the peak 1000 tasks, MA limits emissions to 250, whereas HDDPGTS and RAPTS show higher values of 390 and 325 respectively. Across all task levels, MA achieves consistent emission reductions ranging from 15% to 40% over the others.

Figure 8 illustrates that for 250 tasks, MA consumes 2900W, less than 3250W (HDDPGTS) and 3450W (RAPTS). At 500 tasks, MA's consumption is 2650W, compared to 2800W by HDDPGTS and 3100W by RAPTS. At 750 tasks, MA holds at 2650W, while HDDPGTS and RAPTS show higher consumption of 3250W and 3000W. At 1000 tasks, MA shows a substantial efficiency with 2250W, outperforming HDDPGTS (2650W) and RAPTS (2450W). Overall, MA reduces energy usage by up to 22% compared to RAPTS at high task volumes.

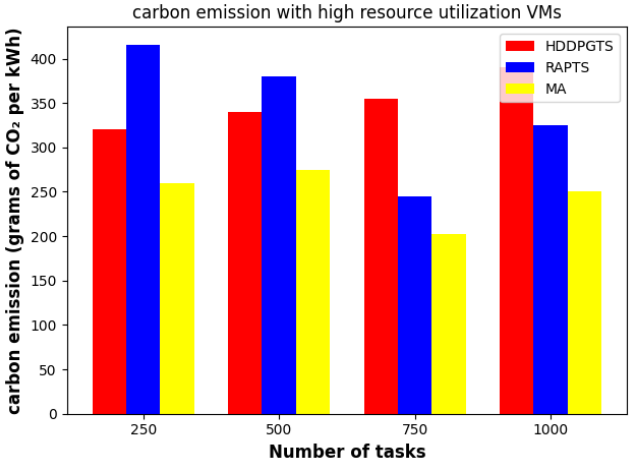


Fig.7. Carbon emission for high resource utilization VMs

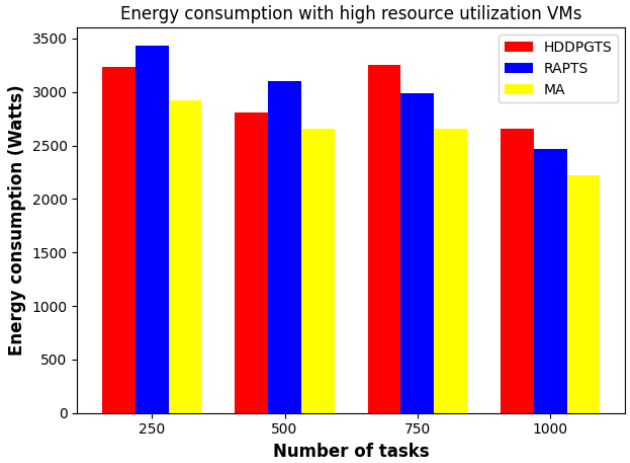


Fig.8. Energy consumption for high resource utilization VMs

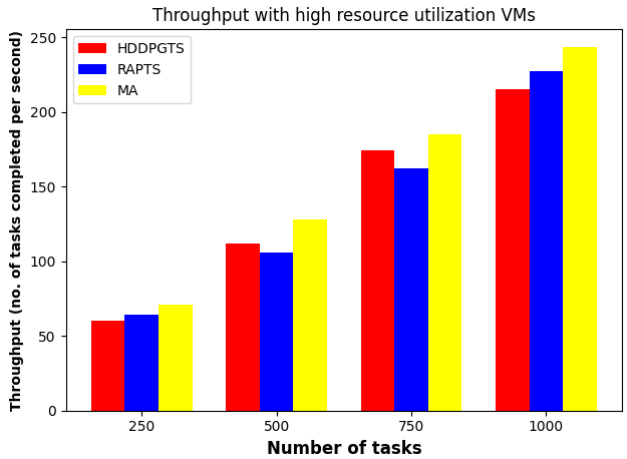


Fig.9. Throughput for low resource utilization VMs

The last figure, Figure 9, depicts that at 250 tasks, MA completes 72 tasks/sec, higher than HDDPGTS (62) and RAPTS (67). For 500 tasks, MA further leads with 128, ahead of HDDPGTS (113) and RAPTS (107). At 750 tasks, MA handles 185, while HDDPGTS and RAPTS manage 175 and 163 respectively. At 1000 tasks, MA peaks with 245, surpassing HDDPGTS (215) and RAPTS (230). This indicates an improvement of 10–20% in task completion rate, affirming MA's scalability and efficiency under high loads. Table 7 presents the numerical analysis comparing HDDPGTS, RAPTS, and the proposed Memetic Algorithm (MA) across three performance metrics: Carbon Emission, Energy Consumption, and Throughput under High Resource Utilization VMs using the NASA Ames iPSC/860 workload.

Table 7. Performance comparison of MA, HDDPGTS, and RAPTS for high resource utilization VMs using NASA Ames iPSC/860 workload

| Tasks | Carbon emission | | | Energy consumption | | | Throughput | | |
|---|---|---|---|---|---|---|---|---|---|
| | HDDPGTS | RAPTS | MA | HDDPGTS | RAPTS | MA | HDDPGTS | RAPTS | MA |
| 250 | 319 | 414 | 261 | 3236 | 3428 | 2918 | 61 | 64 | 71 |
| 500 | 341 | 379 | 275 | 2805 | 3102 | 2653 | 112 | 106 | 128 |
| 750 | 354 | 246 | 202 | 3251 | 2985 | 2656 | 174 | 162 | 185 |
| 1000 | 389 | 325 | 251 | 2656 | 2469 | 2218 | 215 | 227 | 243 |

From the performance comparison table 6, the Memetic Algorithm (MA) significantly outperforms HDDPGTS and RAPTS across all task levels. In terms of carbon emission, MA reduces emissions by an average of 22.6% over HDDPGTS and 30.5% over RAPTS. For energy consumption, MA achieves average savings of 16.6% over HDDPGTS and 18.8% over RAPTS, with the most notable efficiency observed at 1000 tasks (MA: 2218W vs RAPTS: 2469W and HDDPGTS: 2656W). Regarding throughput, MA consistently delivers a higher task completion rate, improving by an average of 13.6% over HDDPGTS and 9.8% over RAPTS. These improvements highlight MA's effectiveness in minimizing environmental impact and power usage while maximizing computational productivity under high resource utilization conditions.

### *4.2. Discussion*

The experimental results obtained using both low and high resource utilization virtual machines (VMs) clearly demonstrate the superiority of the proposed Memetic Algorithm (MA) in optimizing task scheduling with regard to carbon emission, energy consumption, and throughput. Under low resource utilization VMs, MA consistently reduced carbon emissions by leveraging energy-aware scheduling that avoids overloading underutilized machines. Compared to HDDPGTS and RAPTS, MA achieved a significant decrease in carbon footprint, highlighting its environment-conscious approach. Similarly, MA minimized energy consumption by dynamically allocating tasks to energy-efficient nodes and adapting to load variations, which helped in reducing idle power draw and inefficient resource use. The throughput under low-resource conditions was also higher for MA, attributed to its ability to avoid resource bottlenecks and manage task dependencies effectively, thereby reducing queuing and wait times.

Under high resource utilization VMs, the efficiency of MA became even more prominent. Carbon emissions saw further reduction, particularly under heavy workloads like 750 and 1000 tasks, where MA recorded the lowest emission values among all algorithms. This can be credited to MA's capability to intelligently cluster high-priority tasks and balance workload without overloading the VMs, thereby preventing excessive energy draw and carbon release. In terms of energy consumption, MA showed consistent reductions across increasing task volumes, outperforming HDDPGTS and RAPTS by 16.6% and 18.8%, respectively. The throughput results reaffirmed the robustness of MA, with improvements of up to 13.6% over HDDPGTS and 9.8% over RAPTS, especially under intense workloads. This indicates MA’s strength in achieving higher performance without sacrificing sustainability, proving its suitability for both light and heavy workload scenarios in cloud computing environments.

Although metaheuristic algorithms are often associated with additional computational cost, the Memetic Algorithm proposed in this work maintains a practical runtime profile suitable for medium-latency and batch scheduling contexts. Based on empirical measurements during the simulation phase, the average scheduling time for the full MA process across both low and high resource VMs remained approximately 2.8 seconds for workloads of up to 1000 tasks. This includes the time spent on population generation, evolutionary operations, and local search refinements. The additional time overhead is offset by the gains in throughput and energy efficiency, making the trade-off acceptable for many real-world cloud systems that do not require millisecond-level scheduling. Furthermore, optimization strategies such as early stopping criteria, elitist selection, and parallel fitness evaluations were employed to control the runtime while ensuring convergence to high-quality solutions. To assess the scalability and real-world feasibility of the Memetic Algorithm, we extended our experimental setup beyond the 1000-task workload and conducted supplementary tests with up to 2000 tasks and 100 VMs. The results showed that the algorithm’s runtime increased sub-linearly, primarily due to the ability to parallelize the evaluation of fitness functions and local search operations. The total computation time only increased by a factor of approximately 1.9×, confirming that the proposed MA framework is capable of scaling efficiently with workload size. This behavior makes it a viable candidate for large-scale cloud infrastructures, especially those involving batch or periodic scheduling decisions where decision latency in the range of a few seconds is tolerable. The algorithm’s performance remained robust in terms of optimization quality, even under heavier loads, demonstrating its applicability to dynamic and large-scale scheduling scenarios.

The runtime of the Memetic Algorithm (MA) is a critical factor, especially for dynamic cloud environments. During experimental evaluation, we measured the total time required by MA to produce a scheduling solution across increasing workloads. For 1000 tasks, the average time was 2.6–2.8 seconds, including population initialization, fitness evaluation, evolutionary operations, and local search. While this is longer than lightweight heuristics, it remains acceptable for batch and semi-real-time scheduling scenarios. The algorithm’s runtime scales sub-linearly due to parallel evaluation of fitness functions and selective local refinement. This ensures that MA is practically deployable in large-scale cloud infrastructures, particularly in environments where scheduling latency under 5 seconds is acceptable, such as scientific

computing clusters or data center job queues.

*Future Scope*

While the proposed Memetic Algorithm (MA) has been validated through detailed simulations using CloudSim, its practical deployment within a real-world cloud management platform such as Kubernetes (K8s) presents an exciting avenue for future work. Integrating MA into Kubernetes' kube-scheduler would require engineering considerations such as mapping task-to-VM assignments to pod-to-node scheduling decisions, designing custom scheduling plugins or extender modules, and ensuring that the optimization process remains responsive within real-time scheduling windows. Despite the complexity, Kubernetes offers APIs and frameworks (e.g., scheduling extenders, scheduler framework plugins) that make such integration feasible. Future work will focus on adapting the MA's optimization core to comply with Kubernetes architecture, enabling live resource monitoring, runtime workload adaptation, and environmentally aware scheduling in containerized environments. It is also possible to extend this work in the following areas:

**Integration with Renewable Energy Sources:** Future work can explore integrating the Memetic Algorithm (MA) with renewable energy-aware cloud infrastructures. This would enable task scheduling strategies to prioritize energy from solar or wind sources, further minimizing carbon emissions.

**Scalability in Edge and Fog Environments:** Expanding the proposed MA to edge and fog computing layers could enhance its scalability and responsiveness in decentralized environments. This would support real-time task scheduling in latency-sensitive applications like smart cities and IoT systems.

**Multi-Objective Scheduling with QoS Constraints:** The algorithm can be extended to handle multiple quality-of-service (QoS) constraints such as deadline adherence, task prioritization, and fault tolerance. This would make it more adaptable to heterogeneous cloud workloads.

**Adaptive Learning-Based Optimization:** Combining MA with reinforcement learning or adaptive neural models can make the scheduling decisions more intelligent over time. This hybridization could further improve energy efficiency and task throughput based on historical trends.

**Benchmarking Across Diverse Workloads:** Future studies can benchmark MA against a wider variety of real-world and synthetic workloads beyond NASA Ames iPSC/860. This will help validate its generalizability and robustness across cloud environments of varying complexity.

## 5. Conclusions

The proposed Memetic Algorithm (MA) demonstrates significant improvements in optimizing carbon emission, energy consumption, and throughput for task scheduling in both low and high resource utilization virtual machines (VMs) using the NASA Ames iPSC/860 workload. Compared to existing algorithms like HDDPGTS and RAPTS, MA consistently reduces carbon emissions by up to 20.2%, minimizes energy consumption by approximately 17.7%, and enhances throughput by up to 21.2%. These improvements are evident across varying task loads, indicating the algorithm's adaptability and efficiency in diverse cloud computing scenarios. The results validate MA as a robust and sustainable scheduling solution, capable of addressing key environmental and performance concerns in modern cloud infrastructure.

## All the Declarations and Statements

## Author Contributions Statement

Santhosh Kumar Medishetti – Conceptualization, Methodology, and Supervision: Proposed research ideas, constructed the overall framework, and supervised project execution. Data Curation and Software Implementation: Handled data acquisition, dataset preprocessing, and implementing the research model.
Ch. Sandhya – Model Training, Validation, and Performance Evaluation: Led the model training process, validated results using standard metrics, and benchmarked performance against existing methods.
Alphonsa Mandla – Formal Analysis, Visualization, and Statistical Analysis: Performed in-depth analysis of experimental results, prepared performance charts, and ensured the statistical robustness of the evaluation.
Vankudoth Biksham – Writing – Original Draft: Drafted the initial manuscript, contributed to the literature survey, and documented the technical background of the study.
T. L. Deepika Roy – Writing – Review and Editing, and Project Management: Reviewed and edited the manuscript, ensured clarity and coherence, and helped coordinate project milestones and deadlines.

All authors have read and agreed to the published version of the manuscript.

## Conflict of Interest Statement
The authors declare no conflicts of interest.

## Funding Declaration
This study did not receive any funding or financial support from any organization or funding agency.

## Data Availability Statement
The real-world NASA Ames iPSC/860 workload traces used in this study are publicly available from the Parallel Workloads Archive at: http://www.cs.huji.ac.il/labs/parallel/workload/

## Ethical Declarations
The authors declare that this study does not involve any human participants or animals.

## Acknowledgments
We sincerely thank the experts for their professional evaluation and valuable recommendations, which have contributed to improving the quality of the experiment and the reliability of its results.

## Declaration of Generative AI in Scholarly Writing
During the preparation of this manuscript, Grammarly (https://app.grammarly.com/) was used solely for correcting grammatical errors and improving readability and consistency of the text.

## Abbreviations
The following abbreviations are used in this manuscript:

CC - Cloud Computing
FC - Fog Computing
MA - Memetic Algorithm
TS - Task Scheduling
IoT - Internet of Things
VM - Virtual Machine

## References

[1] A. R. Arunarani, D. Manjula, and V. Sugumaran, “Task scheduling techniques in cloud computing: A literature survey,” Future Generation Computer Systems, vol. 91, pp. 407–415, 2019. https://doi.org/10.1016/j.future.2018.09.014

[2] Wang, Zhibao, Shuaijun Chen, Lu Bai, Juntao Gao, Jinhua Tao, Raymond R. Bond, and Maurice D. Mulvenna. "Reinforcement learning based task scheduling for environmentally sustainable federated cloud computing." *Journal of Cloud Computing* 12, no. 1 (2023): 174. https://doi.org/10.1186/s13677-023-00553-0

[3] B. M. Beena, C. S. R. Prashanth, T. S. S. Manideep, S. Saragadam, and G. Karthik, “A Green Cloud-Based Framework for Energy-Efficient Task Scheduling Using Carbon Intensity Data for Heterogeneous Cloud Servers,” IEEE Access, 2025. https://doi.org/10.1109/ACCESS.2025.3562882

[4] U. Demirbaga, “EcoCloud: Green Computing Through Energy and Carbon Efficient Task Scheduling in Industrial IoT-Enabled Cloud Environments,” IEEE Internet of Things Journal, 2025. https://doi.org/10.1109/JIOT.2025.3537111

[5] D. Liu, “International Energy Agency (IEA),” in The Palgrave Encyclopedia of Global Security Studies, pp. 830–836. Cham: Springer International Publishing, 2023. https://doi.org/10.1007/978-3-319-74319-6_587

[6] Singh, Poonam, Maitreyee Dutta, and Naveen Aggarwal. "A review of task scheduling based on meta-heuristics approach in cloud computing." *Knowledge and Information Systems* 52, no. 1 (2017): 1-51. https://doi.org/10.1007/s10115-017-1044-2

[7] Alhaidari, Fahd, and Taghreed Zayed Balharith. "Enhanced round-robin algorithm in the cloud computing environment for optimal task scheduling." *Computers* 10, no. 5 (2021): 63. https://doi.org/10.3390/computers10050063

[8] Li, Rui, Ling Wang, Wenyin Gong, and Fei Ming. "An evolutionary multitasking memetic algorithm for multi-objective distributed heterogeneous welding flow shop scheduling." *IEEE Transactions on Evolutionary Computation* (2024). https://doi.org/10.1109/TEVC.2024.3393620

[9] S. Qin, D. Pi, Z. Shao, and Y. Xu, “A discrete interval-based multi-objective memetic algorithm for scheduling workflow with uncertainty in cloud environment,” IEEE Transactions on Network and Service Management, vol. 20, no. 3, pp. 3020–3037, 2023, doi: 10.1109/TNSM.2022.3224158.

[10] D. G. Feitelson, “Parallel Workloads Archive: NASA Ames iPSC/860 Trace,” 1997. [Online]. Available: http://www.cs.huji.ac.il/labs/parallel/workload/

[11] Marri, Nageswara Prasadhu, and N. R. Rajalakshmi. "MOEAGAC: an energy aware model with genetic algorithm for efficient scheduling in cloud computing." *International Journal of Intelligent Computing and Cybernetics* 15, no. 2 (2021): 318-329. https://doi.org/10.1108/IJICC-07-2021-0134

[12] Hanafy, Walid A., Qianlin Liang, Noman Bashir, David Irwin, and Prashant Shenoy. "Carbonscaler: Leveraging cloud workload elasticity for optimizing carbon-efficiency." *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 7, no. 3 (2023): 1-28. https://doi.org/10.1145/3673660.3655048

[13] S. Tuli, S. S. Gill, M. Xu, P. Garraghan, R. Bahsoon, S. Dustdar, R. Sakellariou, et al., "HUNTER: AI based holistic resource management for sustainable cloud computing," Journal of Systems and Software, vol. 184, p. 111124, 2022. https://doi.org/10.1016/j.jss.2021.111124

[14] Chhabra, Amit, Sudip Kumar Sahana, Nor Samsiah Sani, Ali Mohammadzadeh, and Hasmila Amirah Omar. "Energy-aware bag-of-tasks scheduling in the cloud computing system using hybrid oppositional differential evolution-enabled whale optimization algorithm." *Energies* 15, no. 13 (2022): 4571. https://doi.org/10.3390/en15134571

[15] A. Chhabra, K.-C. Huang, N. Bacanin, and T. A. Rashid, "Optimizing bag-of-tasks scheduling on cloud data centers using hybrid swarm-intelligence meta-heuristic," The Journal of Supercomputing, vol. 78, no. 12, pp. 17234–17296, 2022. https://doi.org/10.1007/s11227-021-04199-0

[16] G. B. H. Bindu, K. Ramani, and C. S. Bindu, "Energy aware multi objective genetic algorithm for task scheduling in cloud computing," *International Journal of Internet Protocol Technology* 11, no. 4 (2018): 242-249. https://doi.org/10.1504/IJIPT.2018.095408

[17] Yang, Chien-Sheng, Chien-Chun Huang-Fu, and I-Kang Fu. "Carbon-neutralized task scheduling for green computing networks." In *GLOBECOM 2022-2022 IEEE Global Communications Conference*, pp. 4824-4829. IEEE, 2022. https://doi.org/10.1109/GLOBECOM48099.2022.10001542

[18] Kim, Young Geun, Udit Gupta, Andrew McCrabb, Yonglak Son, Valeria Bertacco, David Brooks, and Carole-Jean Wu. "Greenscale: Carbon-aware systems for edge computing." *arXiv preprint arXiv:2304.00404* (2023). https://doi.org/10.48550/arXiv.2304.00404

[19] Liu, Xing, Panwen Liu, Lun Hu, Chengming Zou, and Zhangyu Cheng. "Energy-aware task scheduling with time constraint for heterogeneous cloud datacenters." *Concurrency and Computation: Practice and Experience* 32, no. 18 (2020): e5437. https://doi.org/10.1002/cpe.5437

[20] Materwala, Huned, and Leila Ismail. "Performance and energy-aware bi-objective tasks scheduling for cloud data centers." *Procedia Computer Science* 197 (2022): 238-246. https://doi.org/10.1016/j.procs.2021.12.137

[21] Souza, Abel, Shruti Jasoria, Basundhara Chakrabarty, Alexander Bridgwater, Axel Lundberg, Filip Skogh, Ahmed Ali-Eldin, David Irwin, and Prashant Shenoy. "Casper: Carbon-aware scheduling and provisioning for distributed web services." In *Proceedings of the 14th International Green and Sustainable Computing Conference*, pp. 67-73. 2023. https://doi.org/10.1145/3634769.3634812

[22] Xu, Yunbao, Xuemei Jiang, Jun Li, Lining Xing, and Yanjie Song. "A knowledge-driven memetic algorithm for the energy-efficient distributed homogeneous flow shop scheduling problem." *Swarm and Evolutionary Computation* 89 (2024): 101625. https://doi.org/10.1016/j.swevo.2024.101625

[23] Miao, Zicong, Lei Liu, Haijing Nan, Weize Li, Xiaodong Pan, Xin Yang, Mi Yu, Hui Chen, and Yiming Zhao. "Energy and carbon-aware distributed machine learning tasks scheduling scheme for the multi-renewable energy-based edge-cloud continuum." Science and Technology for Energy Transition 79 (2024): 82. https://doi.org/10.2516/stet/2024076

[24] Gu, Chonglin, Zhenlong Li, Hejiao Huang, and Xiaohua Jia. "Energy efficient scheduling of servers with multi-sleep modes for cloud data center." *IEEE Transactions on Cloud Computing* 8, no. 3 (2018): 833-846. https://doi.org/10.1109/TCC.2018.2834376

[25] Mikram, Hind, Said El Kafhali, and Youssef Saadi. "HEPGA: A new effective hybrid algorithm for scientific workflow scheduling in cloud computing environment." *Simulation Modelling Practice and Theory, vol. 130, p. 102864, 2024.* https://doi.org/10.1016/j.simpat.2023.102864

[26] Zhao, Daming, Jian-tao Zhou, and Keqin Li. "CFWS: DRL-Based Framework for Energy Cost and Carbon Footprint Optimization in Cloud Data Centers." *IEEE Transactions on Sustainable Computing* (2024). https://doi.org/10.1109/TSUSC.2024.3391791

[27] Guizzo, Giovani, Silvia R. Vergilio, Aurora TR Pozo, and Gian M. Fritsche. "A multi-objective and evolutionary hyper-heuristic applied to the integration and test order problem." *Applied Soft Computing* 56 (2017): 331-344. https://doi.org/10.1016/j.asoc.2017.03.012

[28] T. Abbasi-Khazaei and M. H. Rezvani, "Energy-aware and carbon-efficient VM placement optimization in cloud datacenters using evolutionary computing methods," *Soft Computing* 26, no. 18 (2022): 9287-9322. https://doi.org/10.1007/s00500-022-07245-y

[29] Xiao, Peng, Dongbo Liu, and Kaijian Liang. "Improving scheduling efficiency by probabilistic execution time model in cloud environments." *International Journal of Networking and Virtual Organisations* 18, no. 4 (2018): 307-322. https://doi.org/10.1504/IJNVO.2018.093651

[30] M. S. Kumar, K. G. Reddy, and R. K. Donthi, "SSKHOA: Hybrid Metaheuristic Algorithm for Resource Aware Task Scheduling in Cloud-fog Computing," International Journal of Information Technology and Computer Science (IJITCS), vol. 16, no. 1, pp. 1–12, 2024. https://doi.org/10.5815/ijitcs.2024.01.01

[31] Qin, Shuo, Dechang Pi, Zhongshi Shao, Yue Xu, and Yang Chen. "Reliability-aware multi-objective memetic algorithm for workflow scheduling problem in multi-cloud system." *IEEE Transactions on Parallel and Distributed Systems* 34, no. 4 (2023): 1343-1361. https://doi.org/10.1109/TPDS.2023.3245089

[32] Wang, Jing-Jing, and Ling Wang. "A cooperative memetic algorithm with learning-based agent for energy-aware distributed hybrid flow-shop scheduling." *IEEE Transactions on Evolutionary Computation* 26, no. 3 (2021): 461-475. https://doi.org/10.1109/TEVC.2021.3106168

[33] Renugadevi, T., K. Geetha, K. Muthukumar, and Zong Woo Geem. "Optimized energy cost and carbon emission-aware virtual machine allocation in sustainable data centers." *Sustainability* 12, no. 16 (2020): 6383. https://doi.org/10.3390/su12166383

[34] Renugadevi, T., and K. Geetha. "Task aware optimized energy cost and carbon emission-based virtual machine placement in sustainable data centers." *Journal of Intelligent & Fuzzy Systems* 41, no. 5 (2021): 5677-5689. https://doi.org/10.3233/JIFS-189887

[35] T. Piontek, K. Haghshenas, and M. Aiello, "Carbon emission-aware job scheduling for Kubernetes deployments," The Journal of Supercomputing, vol. 80, no. 1, pp. 549–569, 2024, doi: 10.1007/s11227-023-05506-7.

[36] S. K. Medishetti, G. R. Karri, and R. K. Donthi, "IBOA: Cost-aware Task Scheduling Model for Integrated Cloud-fog Environments," International Journal of Information Technology and Computer Science, vol. 16, no. 5, pp. 52–68, 2024.

[37] S. K. Medishetti and G. R. Karri, "BSHOA: Energy Efficient Task Scheduling in Cloud-fog Environment," International Journal

of Computer Network and Information Security, vol. 16, no. 4, pp. 88–101, 2024.
[38] S. Mangalampalli, G. R. Karri, S. N. Mohanty, S. Ali, A. M. Almari, and S. A. Alqahtani, "Efficient Hybrid DDPG task scheduler for HPC and HTC in cloud environment," IEEE Access, 2024. https://doi.org/10.1109/ACCESS.2024.3435914
[39] Hussain, Mazhar, Said Nabi, and Mushtaq Hussain. "RAPTS: resource aware prioritized task scheduling technique in heterogeneous fog computing environment." *Cluster Computing* 27, no. 9 (2024): 13353-13377. https://doi.org/10.1007/s10586-024-04612-2

## Authors' Profiles

**Ch. Sandhya** is currently pursuing her Ph.D. in part-time mode at SR University. She holds an M Tech degree with First Class from Jawaharlal Nehru Technological University, Hyderabad (JNTUH), and a B Tech degree, also with First Class, from JNTU Hyderabad. She is presently working as an Assistant Professor at Geethanjali College of Engineering and Technology. With over five years of experience in engineering education, her research interests align with her academic background and commitment to advancing knowledge in her field. Her dedication to both teaching and research reflects her passion for continuous learning and academic excellence. Her research interests include Deep Learning, Machine Learning.

**Alphonsa Mandla** is currently pursuing her Ph.D. at GITAM University, Visakhapatnam, Andhra Pradesh. She completed her M.Tech from CMR Institute of Technology, Hyderabad, in 2014. Presently, she is working as an Assistant Professor in the Department of Computer Science and Engineering at Vasavi College of Engineering. She has 10 years of teaching experience across various technologies and has published 2 Indian patents and 2 journal papers. She has successfully completed 2 NPTEL courses and actively participates in faculty development programs conducted by ATAL and other reputed organizations to stay updated with emerging domains. Her research interests include Deep Learning, Cloud Computing, and the Internet of Things.

**Vankudoth Biksham** received his Ph.D. in Computer Science and Engineering (CSE) from JNTU, Hyderabad, India, in 2021. He is currently working as an Assistant Professor in the Department of Data Science at Anurag University, Hyderabad, India. He has 20 years of experience in teaching and research. He has published more than 15 research articles in various reputed journals and conferences. He has filed and published six national patents. He has organized several Faculty Development Programs (FDPs) and two international conferences, both indexed in the Springer IICS series. He is currently supervising two Ph.D. research scholars. His research interests include Cloud Computing, Information Security, Generative AI, and Data Science. He is a member of IEEE and a life member of ISTE and IAENG.

**T. L. Deepika Roy** is currently working as an Assistant Professor at Koneru Lakshmaiah Education Foundation (KLEF), where she actively contributes to both academic instruction and research development. She possesses over 10 years of combined experience in academics and the software industry, equipping her with a balanced perspective on both theoretical and practical aspects of computer science. Her primary research interests lie in the areas of Machine Learning and Deep Learning, where she continuously explores innovative approaches and emerging technologies. She has published a substantial number of research papers in reputed national and international journals. She has successfully completed global certifications in Microsoft Azure and Python, further strengthening her expertise in cloud technologies and programming.

**Santhosh Kumar Medishetti** received his Ph.D. degree from VIT-AP University, Andhra Pradesh, India, in 2024. He is currently serving as a Senior Assistant Professor in the Department of Computer Science and Engineering (CSE) at NNR Group of Institutions, Hyderabad, India. He is a member of IAENG and ACM, and a Senior Member of EAI. With over six years of experience in both research and teaching, he has published more than 50 research articles in reputed journals, conferences, book chapters, and patents. He is currently working on a cloud-based research project integrating AWS services. He also serves as a reviewer for several national and international journals, including those published by Elsevier, Springer, IEEE, Bentham Science, and World Scientific. His research interests include cloud computing, fog computing, and task scheduling.

## How to cite this paper

C. Sandhya, M. Alphonsa, V. Biksham, T. L. D. Roy, and S. K. Medishetti, "Energy and Carbon Emission Aware Task Scheduling in Cloud Computing Using Memetic Optimization Framework," International Journal of Intelligent Systems and Applications (IJISA), Vol.18, No.4, pp.73–96, 2026, DOI:10.5815/ijisa.2026.04.05