IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 15, No. 3, June 2026, pp. 2723~2735
ISSN: 2252-8938, DOI: 10.11591/ijai.v15.i3.pp2723-2735

Makespan and energy-aware workflow scheduler for heterogeneous cloud computing platform

Rashmi Kambalapalli Anjaneya Reddy[1], Vikas Reddy Shivaram Reddy[2]
[1]Department of Computer Science and Engineering, SJC Institute of Technology, Visvesvaraya Technological University, Belagavi, India
[2]Department of Computer Science and Engineering (Artificial Intelligence and Machine Learning), SJC Institute of Technology, Visvesvaraya Technological University, Belagavi, India

Article Info

Article history:

Received Mar 5, 2025
Revised Feb 9, 2026
Accepted Apr 22, 2026

Keywords:

Directed acyclic graph
Dynamic voltage and frequency scaling
Heterogenous cloud
MEAWS
Makespan optimization
Multi-core architecture
scientific workflow scheduling

ABSTRACT

Scientific workflows, typically modelled as complex directed acyclic graphs (DAGs), are increasingly executed on heterogeneous cloud platforms to achieve high performance and scalability. However, as workflow sizes grow, energy consumption, and operational cost have become critical concerns, especially under global carbon-emission constraints. Although dynamic voltage and frequency scaling (DVFS) offers significant potential for energy savings, existing workflow scheduling methods fail to fully exploit heterogeneous processors that contain both high-performance and energy-efficient cores, resulting in suboptimal makespan and energy utilization. To address this gap, the makespan and energy-aware workflow scheduler (MEAWS) is proposed as a multi-core DVFS-enabled scheduling framework designed to optimize both execution time and energy consumption in heterogeneous cloud environments. Extensive simulations using scientific workflows demonstrate that MEAWS reduces makespan by up to 88.75% and 70.4%, and lowers energy usage by 41.59% and 47.15% when compared with reliable and efficient workflow scheduling (REWS) and multi-objective workflow scheduling (MOWS). These improvements highlight the effectiveness of MEAWS in enhancing the sustainability and efficiency of scientific workflow execution.

Corresponding Author:

Rashmi Kambalapalli Anjaneya Reddy
Department of Computer Science and Engineering, SJC Institute of Technology
Visvesvaraya Technological University
Belagavi, India
Email: rashmika_12@rediffmail.com

1. INTRODUCTION

Scientific workflows, commonly characterized as directed acyclic graphs (DAGs), have become an essential computational paradigm for scientists and industries handling complex resource and data-driven computational studies [1]. These DAG workflows with strong interdependencies facilitate the execution of complex computational tasks that involve different operations such as collection, storage, aggregation, analytics, and modelling and simulation studies. Efficient workflow task scheduling is a fundamental aspect of scientific workflows, ensuring optimal resource management and organized computation [2], [3]. The increasing dependence on scientific workflows has necessitated the implementation of cloud computing to enhance computational efficiency [4]. Cloud platforms provide centralized resource provisioning with flexible and scalable resource management, which makes them appropriate for managing scientific workflows [5]. Nonetheless, significant computational demands pose challenges, including resource limitations, performance bottlenecks, and excessive energy usage [6].

To address these limitations, multi-cloud computational systems are being utilized, improving resource availability across diverse cloud environments [7]. Despite their advantages, multi-cloud computational systems often struggle with latency issues, particularly in realistic computational scenarios [8]. In contrast, edge computing [9] leverages decentralized resources closer to data sources, significantly reducing latency and enhancing real-time processing capabilities. Integrating dynamic resource optimization within multi-core heterogeneous cloud computing further enhances workflow performance. The application of virtual function optimization enables flexible resource allocation, leading to improvements in energy efficiency, reliability, and overall resource utilization. Recently, work to reduce latency was exploited using edge-based computing design [10]; further, a multi-core heterogeneous system [11] has been explored to improve resource utilization. However, all these methods have failed to consider optimizing the energy-makespan deadlines constraint of scientific workflows [7], [12].

A comprehensive analysis of existing scheduling models is presented in section 2, highlighting key research gaps. One primary research challenge is that current scientific workflow scheduling methods typically consider multi-objective [13], [14] parameters such as makespan, cost, and energy efficiency, but focus on either homogeneous or heterogeneous cloud platforms. Scientific workflows often involve stringent deadlines and varying resource requirements across different DAG levels. Failure to execute a sub-task at any stage can result in complete task failure. Additionally, existing scheduling approaches do not effectively identify optimal frequency pairs considering heterogeneous systems, limiting overall performance. Consequently, ensuring deadline-compliant task scheduling in heterogeneous computational frequency-core pairs remains a complex challenge [14].

The motivation for this research stems from the lack of prior multi-objective scheduling strategies [13], [14]. Specifically designed for heterogeneous computational systems composed of both smaller-core and larger-cores. The heterogeneous computation frequency-core architectures [13] introduces unique challenges, including distinct energy consumption models, task partitioning complexities, and the need for scheduling optimization that balances energy consumption and makespan while meeting strict deadlines [14]. This necessitates the development of novel scheduling mechanisms that minimize energy usage while ensuring the timely completion of scientific workflows. To address these challenges, this study proposes a novel makespan and energy-aware workflow scheduler (MEAWS) technique. MEAWS employs dynamic voltage and frequency scaling (DVFS) to reduce energy consumption within heterogeneous systems. The MEAWS model further incorporates an optimization mechanism to determine the most efficient frequency pairs, ensuring task deadlines are met with minimal energy expenditure.

The contribution of this work is as follows: this study introduces an innovative scheduler tailored for executing scientific workflows within a heterogeneous system having different frequency pairs. Unlike previous research, this work uniquely considers the execution of Inspiral and CyberShake scientific workflows in heterogeneous systems. A new energy and makespan model employing DVFS is developed specifically for heterogeneous frequency-core, aiding in better utilization of system resources. The MEAWS algorithm dynamically selects optimal frequency pairs for workflow subtasks, ensuring a lesser makespan and minimal energy usage while adhering to task deadlines. Extensive simulation-based validation demonstrates that MEAWS significantly outperforms existing models, such as the reliability enhancement strategy for workflow scheduling (RESWS), and the multi-objective workflow scheduling (MOWS). It achieves notable reductions in makespan and energy consumption for data-compute-intensive scientific workflows.

This paper organizes as follows. In section 2, different scientific workflow scheduling presented employing cloud-based computational systems. In section 3, working of MEAWS is presented. In section 4, the results attained using MEAWS, another current scheduler, are presented. Section 5 presents the significance and future research direction.

2. RELATED WORK

Workflow scheduling has evolved significantly with the rapid expansion of edge–cloud infrastructures, heterogeneous compute platforms, and the increasing complexity of scientific and internet of things (IoT)-driven applications. Recent research highlights a clear shift toward multi-objective optimization, where energy efficiency, performance, budget constraints, reliability, and fault-tolerance must be simultaneously addressed. The works studied how evolutionary algorithms, deep reinforcement learning (DRL), hybrid heuristics, and adaptive resource management techniques are adopted to improve the workflow scheduling performance.

Lu *et al.* [6] propose a scheduler capable of optimizing both energy consumption and performance by dynamically managing virtual network resources. Their work underscores that workflow bottlenecks now occur not only at the compute level but also across the network fabric, a concern amplified by latency-sensitive

IoT applications. This idea is echoed in broader multi-cloud and network functions virtualization (NFV) environments, where Hossain *et al.* [15] introduce energy-aware server farm scaling strategies with built-in reliability constraints. Their work demonstrates that heterogeneous network infrastructures require adaptive scaling policies that minimize energy usage while sustaining carrier-grade availability.

Budget-aware optimization remains another essential scheduling dimension. Wu *et al.* [9] propose a mechanism combining priority adjustment with critical task optimization to keep execution within user-defined cost boundaries. Their framework dynamically recalibrates task priorities when budget violations are detected, providing a flexible way to maintain performance without overspending. This complements the study by Sreedhara *et al.* [16], who incorporate cost as a primary objective alongside energy and reliability within a multi-objective evolutionary setting for multi-cloud platforms. Their results show that multi-cloud environments, while resource-rich, require careful inter-cloud data transfer management to avoid escalating costs.

The surge in workflow scale and complexity has stimulated the use of multi-objective evolutionary algorithms (MOEAs). Pujar *et al.* [17] present a critical-task-driven evolutionary approach tailored for large-scale workflows, effectively reducing makespan through focused optimization of bottleneck tasks. Similarly, Yang *et al.* [18] adopt particle swarm optimization to balance time and energy in fog-based infrastructures, demonstrating that swarm intelligence remains competitive for environments with limited resources and dynamic topology changes. The importance of MOEAs is also seen [19], who deploy an enhanced differential evolution method for fog computing. Their algorithm shows improved trade-offs among energy, cost, and deadline constraints, particularly for real-time industrial workflows.

In parallel, DRL has emerged as a dominant methodology enabling adaptive and context-aware workflow decisions. Zhao *et al.* [14] propose a DRL-based priority scheduler capable of handling multiple objectives simultaneously, offering significant improvements in response time and energy usage through reward-driven learning. A3C-based workflow scheduling is explored [20], where agent parallelism accelerates policy convergence and enhances decision quality for dynamic workloads. Another extension from the same research group is the energy- and temperature-aware DRL scheduler proposed in [21], which integrates thermal characteristics of cloud servers into decision-making. This direction is particularly relevant for future high-density data centers where thermal hotspots considerably affect system reliability and energy consumption.

Continuous scheduling in heterogeneous environments is addressed [22], who utilize DRL to optimize workflow execution without discretizing resource states. Their method is highly relevant for multi-tenant systems where workflow submissions arrive unpredictably. In a similar line, Perozzi *et al.* [23] explore DRL-based resource allocation in edge–cloud setups, demonstrating that multi-objective utilities—covering energy, makespan, and resource consumption—can be jointly optimized when agents have visibility across both resource layers.

The growing dependence of IoT systems on edge computing has made fault-tolerance and reliability indispensable. Breitfuss *et al.* [24] design a fault-tolerant scheduler that accounts for both energy budgets and stringent deadlines across the edge–cloud continuum. Their method introduces adaptive redundancy strategies that activate only when necessary, reducing unnecessary energy overhead. Reliability as a core scheduling attribute is also emphasized [25], who propose a hybrid multistage scheduling method that aims to maintain reliability while minimizing energy expenditure in IoT workloads. Tang *et al.* [26] similarly explore reliability enhancement in large cloud infrastructures, presenting strategies that limit energy consumption while preserving high workflow completion success rates.

Adaptive provisioning and multi-workflow optimization have gained traction as well. Wang *et al.* [27] introduce a coalition reinforcement learning framework enabling cloud systems to allocate bundle resources depending on workload characteristics. Their work highlights how cooperative agents can manage resource pools more effectively than isolated schedulers, particularly under fluctuating demand. Table 1 shows a study of recent methodologies to enhance the scheduling performance for the execution of scientific workflows in a heterogeneous cloud platform. methods.

Overall, the collected works shown in Table 1 indicate a broad convergence toward intelligent, multi-objective, and resource-aware scheduling frameworks. Classical heuristics continue to play a role [9], especially for small- to medium-sized workflows or environments with limited compute capabilities. However, DRL [20], [21], [23] and hybrid MOEA [19] approaches dominate recent research due to their adaptability, robustness against uncertainty, and capability to simultaneously satisfy energy, performance, budget, and reliability constraints. Despite these improvements, challenges persist in handling computational complexity, dynamic network conditions, and real-time adaptability, motivating the development of more efficient resource- and energy-aware scheduling mechanisms.

Table 1. Comparative study of existing scheduling techniques

Reference	Technique	Environment	Objectives/optimization metrics	Key strengths	Limitations
Lu *et al.* [6]	Dynamic virtual network resource optimization	Edge–cloud	Energy, performance	Efficient network-aware scheduling reduces latency	May require high control overhead
Wu *et al*. [9]	Priority adjustment + critical task optimization	Cloud	Cost, makespan	Budget-constrained scheduling adapts priorities dynamically	Limited scalability for large workflows
Zhao *et al*. [14]	DRL-based multi-objective scheduling	Cloud	Energy, makespan, reliability	Learns adaptive scheduling policies, balances multiple objectives	High training time, computationally intensive
Breitfuss *et al.* [24]	Fault-tolerant scheduling	Edge–cloud	Energy, deadline, cost	Fault-tolerance, energy-efficient	Complexity increases with workflow size
Wu *et al*. [22]	DRL continuous scheduling	Heterogeneous cloud	Makespan, energy	Continuous decision-making adapts to dynamic workloads	Requires accurate state representation
Pujar *et al*. [17]	MOEAs	Cloud	Makespan, resource utilization	Focuses on critical tasks, scalable	Convergence can be slow for very large workflows
Sreedhara *et al.* [16]	MOEA multi-cloud	Multi-cloud	Energy, cost, reliability	Optimizes multiple objectives simultaneously	Inter-cloud communication cost is not fully addressed
He *et al.* [25]	Hybrid multistage scheduling	Cloud-IoT	Energy, reliability	Reliable execution with energy efficiency	Complexity in stage coordination
Chen *et al.* [19]	Improved multi-objective differential evolution	Fog computing	Time, cost, energy	Balances multiple constraints effectively	May not adapt quickly to network changes
Perozzi *et al.* [23]	DRL-based multi-objective resource allocation	Edge–cloud	Energy, makespan, resource utilization	Adaptive and intelligent scheduling	Requires extensive training data
Tang *et al.* [26]	Reliability enhancement strategies	Cloud	Reliability, energy	Enhances task success rates under energy constraints	Limited consideration of execution cost

3. METHOD

This section presents a makespan and energy-aware scheduler for a heterogeneous cloud platform. Subsection 3.1 presents the system model, makespan, and energy-aware scheduler; the section also presents the architecture and complete abstract working flow of the makespan and energy-aware scheduler. Subsection 3.2 discusses the proposed DAG-based scientific workflow model and discusses the novel working energy model of heterogeneous architecture composed of both P-cores and E-core processors. The novel makespan and energy-aware scheduler optimization model is presented in subsection 3.3. Finally, the complete process of the proposed makespan and energy-aware scheduler is provided in subsection 3.4.

3.1. System model

The overall process involved in scheduling scientific workflow in heterogeneous using existing schedulers, such as a reliable and efficient workflow scheduling (REWS), and MOWS, and a proposed MEAWS is provided in the architecture given in Figure 1. Here, the user submits the workflow to the cloud resource administrator (CRA); the CRA puts the task in the workflow queue and analyzes its quality of service (QoS) and service level agreement (SLA) requirements. Then, the CRA uses a scheduler according to the user's requirement; and then computes the resource availability from the workflow resource administrator (WRA); then, task scheduling is performed; the task scheduling is continued till all the tasks in the queue are executed successfully. Then, finally, the result in terms of task execution status, makespan, and energy consumed is communicated to users.

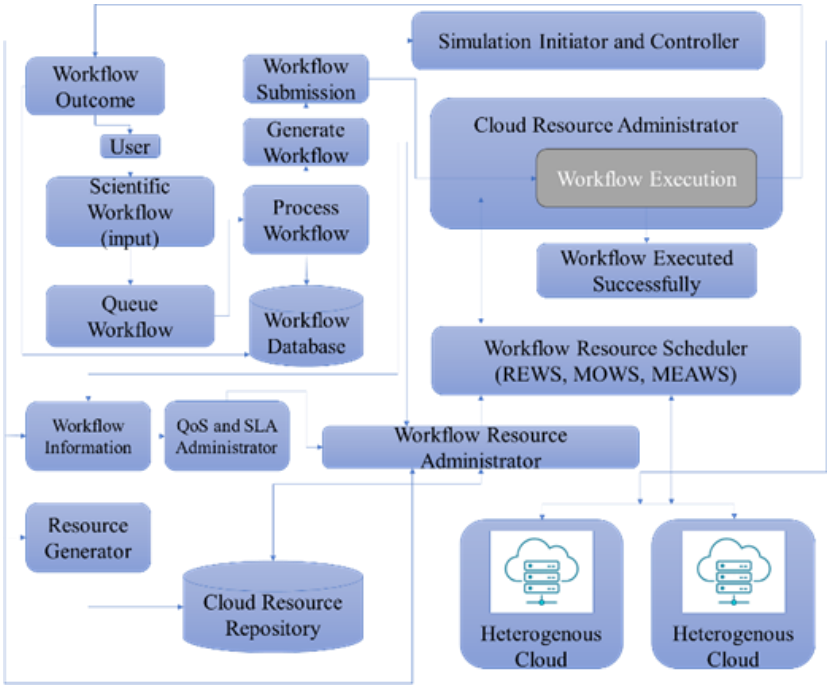


Figure 1. Proposed heterogenous cloud computing architecture for execution of scientific workflows

3.2. Scientific workflow, makespan, and energy model

The scientific workflow is represented as DAG G with connected edge E and vertices as defined in (1) as shown in Figure 2, where the actual subtask is defined by a parameter $X = \{x_1, x_2, \ldots, x_n\}$ and $E = \{e_{jl}\ (s_k, s_m)\}$ defines the dependency between subtasks. A sample representation of G with 10 tasks is given in Figure 2, where it has 4 levels. The level 1 has one task, which is the beginning of the task, the level 2 has 5 tasks, the level 3 has 3 tasks, and level 4 has 1 task, which is the end of the task. Every sub-task x_j with size K_j will require operational frequency F_j. The parameter s_k and s_m define the actual virtual computational node (VCN) allocated to x_j and x_l, respectively. This work considered heterogeneous cores, i.e., the VCN is equipped with both smaller-core or efficiency-core (E-cores) or larger-core/performance-core (P-cores); thus, the makespan is dependent on its processing capability and, therefore, the overall makespan for execution of G is computed in (2).

$$G = \{X, E\} \tag{1}$$

$$G = \mathbb{F}_j \times \mathbb{K}_j \tag{2}$$

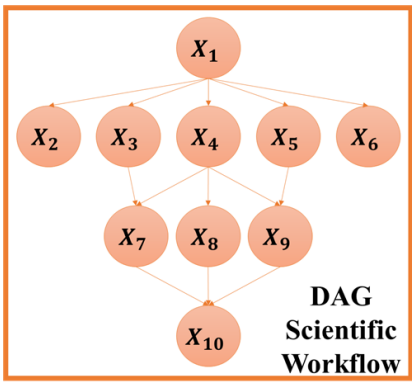


Figure 2. A sample representation of DAG-based scientific workflows

As stated earlier, the makespan is dependent of type of frequency pair and the number of sub-task sizes considered for execution. Therefore, the smaller-core and larger-core clock frequency is designed by parameter $f_j^{(c)}$ and $f'^{(c)}_j$, respectively. The smaller-core frequency $f_j^{(c)}$ is computed in (3). Where $\mathbb{f}_j$ defines smaller-core floating-point capacity within different cycles. The larger-core frequency $f'^{(c)}_j$ is computed in (4).

$$f_j = f_j^{(c)} \times \mathbb{f}_j \tag{3}$$

$$f'_j = f'^{(c)}_j \times \mathbb{f}'_j \tag{4}$$

Where $\mathbb{f}'_j$ defines larger-core floating-point capacity within different cycles. Thus, considering the local gradient function, the task G is partitioned across a smaller-core node j defines as G_j and a larger-core node j defined as G'_j is defined in (5). Thus, considering fine-grained parallelism across both smaller-core and larger-core using (5), the overall makespan maximum time is measured in (6).

$$G_j + G'_j = G, \ \ \forall j \in J \tag{5}$$

$$u'_j = max\left\{\frac{G_j}{f_j}, \frac{G'_j}{f'_j}\right\}, \ \ \forall j \in J \tag{6}$$

Once the makespan is approximated, the work employs a DVFS-based energy model; the energy consumption on both smaller-core and larger-core is obtained in (7). Where the parameter β is measured in $(Watt)(cycle/s)^3$, which are dependent on the kind of processing element being used and $f^{(c)}$ defines the accentual operating frequency of the processing element. Using (7), the energy consumption of smaller-core processors for the corresponding voltage-constrained mode (VCM) J is measured in (8).

$$Q = \beta\left[f^{(c)}\right]^3 \tag{7}$$

$$Q_j^{SC} = \beta_j^{SC}\left(f_j^{(c)}\right)^3 = \mathcal{C}_j f_j^3 \tag{8}$$

Where in (8), $\mathcal{C}_j$ are measured using (9). The energy consumption of larger-core processors for the corresponding VCM J is measured in (10). Where in (10), $\mathcal{G}_j$ are measured using (11). The adoption of the DVFS model in (11) enabled the model to significantly reduce the energy consumption of a smaller-core processor f_j and a larger-core processor f'_j. The parameter $\mathcal{C}_j$ and $\mathcal{G}_j$ used in (8) to (11) enabled the model to meet the task deadline with an energy constraint. Thus, the energy needed to consider makespan $(\frac{G_j}{f_j})$. Executing a task within the given deadline by the smaller-core processor is measured in (12). Similarly, the energy needed to consider makespan $(\frac{G_j}{f_j})$ to execute task within a given deadline by the larger-core processor is measured in (13). Thus, the total energy consumption is measured in (14).

$$\mathcal{C}_j = \frac{\beta_j^{SC}}{\mathbb{f}_j^3} \tag{9}$$

$$Q_j^{LC} = \beta_j^{LC}\left(f_j'^{(c)}\right)^3 = \mathcal{G}_j f_j'^3 \tag{10}$$

$$\mathcal{G}_j = \frac{\beta_j^{LC}}{\mathbb{f}_j'^3}. \tag{11}$$

$$\mathcal{E}_j^{SC} = \mathcal{C}_j G_j f_j^2 \tag{12}$$

$$\mathcal{E}_j^{LC} = \mathcal{G}_j G'_j f_j'^2. \tag{13}$$

$$\mathcal{E}_j^{cmp} = \mathcal{C}_j G_j f_j^2 + \mathcal{G}_j G'_j f_j'^2, \ \ \forall j \in J. \tag{14}$$

The energy model adopted in this study follows widely validated DVFS-based power–frequency relationships used in prior workflow scheduling and processor energy modelling literature [19], [23], [27]. The cubic dependency between operating frequency and dynamic power has been consistently demonstrated through empirical processor measurements in heterogeneous multi-core systems. These empirical results thereby support the applicability of the proposed model parameters β, $\mathcal{C}_j$, and $\mathcal{G}_j$.

3.3. Makespan-energy optimization

As discussed in the previous section, the makespan and energy consumption are computed in (6) and (14), respectively. Next, the work introduces a novel MEAWS scheduler by introducing three variables to represent a heterogeneous architecture. The parameter S defines the processor computational capacity of smaller-core and larger-core put together, S_{SC} defined the processor computational capacity of a smaller-core, and S_{LC}, defines the processor computational capacity of larger-core. Therefore, the makespan for smaller-core systems are obtained in (15).

$$U_{SC} = \frac{G_{SC}}{S_{SC}} \tag{15}$$

Similarly, the makespan for larger-core systems is obtained in (16). Where G_{SC} defines task allocated to a smaller-core system is measured in (17). Where G_{SC} defines task allocated to smaller-core system is measured in (17). Similarly, G_{LC} defines task allocated to larger-core system is measured in (18).

$$U_{LC} = \frac{G_{LC}}{S_{LC}} \tag{16}$$

$$G_{SC} = (1-\gamma) \times G \tag{17}$$

$$G_{LC} = \gamma \times G \tag{18}$$

The parameter γ is used as an optimization parameter, ranging between $0 \leq \gamma \leq 1$ to allocate tasks among the smaller-core and larger-core. The optimal efficiency can be obtained by minimizing energy with minimal makespan by scheduling tasks between smaller-core and larger-core systems in a cooperative manner using (19). Thus, considering cooperative scheduling, optimal efficiency $\gamma_{optimal}$ is obtained in (20). Thus, using (20), the performance for scheduling scientific workload is improved by employing (21). Therefore, using (21), the scheduling with minimal makespan is obtained employing (22). On the other side, the energy needed δ to complete scientific workflow execution can be significantly reduced using (23).

$$U_{SC} = U_{LC} \text{ or } \frac{(1-\gamma)*G}{S_{SC}} = \frac{\gamma \times G}{S_{LC}} \tag{19}$$

$$\gamma_{optimal} = \frac{S_{LC}}{S_{SC}+S_{LC}} \tag{20}$$

$$S = S_{SC} + S_{LC} \tag{21}$$

$$\frac{G}{S_{SC}+S_{LC}} \tag{22}$$

$$\delta = \frac{S_{SC}+S_{LC}}{\varepsilon_j^0+\varepsilon_j^{GPU}+\varepsilon_j^{LC}} \tag{23}$$

Using a proposed heterogeneous architecture composed of smaller-core and larger-core systems together enables the model to work in different frequencies to perform energy optimization by varying the voltage-frequency level. The frequency of a smaller-core is defined by M_{SC} and a larger-core is defined by a parameter M_{LC}. Thus, the summation concerning $M_{SC} \times M_{LC}$ is feasible as soon as $(f_{SC}\ and\ f_{LC})$ are in pairs. Additionally, energy makespan optimal assurance for every pair. Thus, in this work, a matrix $(f_{SC}^*, f_{LC}^*, \gamma^*)$ is constructed in order to enhance the performance efficiency with minimal energy consumption. In this work, according to the arrival scientific workflow, the optimal pair is selected. i) for every pair (f_{SC}, f_{LC}), the MEAWS obtains $S_{max}(f_{SC}, f_{LC})$ and $(f_D, f_H, \gamma_{optimal})$; ii) for the obtained $S_{max}(f_{SC}, f_{LC})$, decide optimal frequency $(f_{SC}^*, f_{LC}^*, \gamma_{optimal}^*)$; iii) for every pair (f_{SC}, f_{LC}), the MEAWS obtains $\delta_{max}(f_{SC}, f_{LC})$ and $(f_{SC}^*, f_{LC}^*, \gamma_{optimal}^*)$; and iv) for the obtained $\delta_{max}(f_{SC}, f_{LC})$, decide the optimal frequency $(f_{SC}^*, f_{LC}^*, \gamma_{optimal}^*)$.

3.4. Algorithm of makespan and energy-aware scheduler

This section discusses the step-by-step process of the working of the MEAWS approach. The approach is applied to a novel heterogeneous system that includes both smaller-core and larger-core processors. These processors differ in storage, memory, and bandwidth, and the overall process is outlined in Algorithm 1.

Algorithm 1. Makespan and energy-aware scheduler approach
1: Start
2: Deployment of heterogeneous systems composed of physical machines, virtual computing nodes with smaller-cores, and larger-cores having bandwidth, storage, and memory.
3: Users submit a scientific workflow represented as DAG G.
4: ∀ incoming scientific workflows G accepted by the CWA do
5: CWA investigates G's QoS and SLA constraint and communicate to CRA
6: CRA divides sub-task X of scientific workflow G to smaller-cores employing (17) and larger-cores employing (18).
7: CRA establishes the optimal pairs employing (20) and (23)
8: End ∀
9: Stop

The deployment of a heterogeneous system is performed. In step 3, the users submit the scientific workflows G having complex dependencies among sub-tasks to the cloud platform. The cloud workflow administrator (CWA) puts the task into queues and analyses the workflow QoS and SLA requirements. Then, the CRA establishes the resource and starts allocating resource pairs, considering smaller-core and larger-core according to the task requirement. The scheduling process is iterated till all the task in the queue become empty. Finally, the simulation is stopped, and the outcome is communicated to the user with the amount of makespan and energy spent. The adoption of the heterogeneous and optimal pairing concept during scheduling significantly aids the model to attain enhanced makespan and energy efficiency using MEAWS as shown through a simulation study in the next section.

4. RESULT AND DISCUSSION

This section presents an experimental evaluation of the proposed MEAWS for executing scientific workflows in a heterogeneous computational system. To evaluate the performance of MEAWS, it is compared with existing scheduling strategies, namely REWS [28] and MOWS [14], [19]. REWS serves as a baseline due to its emphasis on minimizing energy consumption while meeting task deadlines, whereas MOWS targets improved resource utilization alongside reduced makespan and energy use. All scheduling approaches, including MEAWS, are implemented and tested using the CloudSim simulation framework [25], [28], [29]. The Inspiral and CyberShake scientific workflows are employed as benchmark applications [30], with Inspiral presenting high memory and computational demands and CyberShake imposing intensive I/O and processing workloads. These benchmarks allow for robust assessment across diverse computational scenarios. The simulations focus on evaluating key performance indicators, specifically, makespan, and energy consumption. Experiments are conducted on a Windows 11 environment equipped with an Intel Core i7 processor, 16 GB of RAM, and CUDA-enabled GPU with 4 GB of memory. The cloud setup comprises five physical hosts hosting 50 virtual machines, representing a realistic execution environment. Table 2 summarizes the simulation parameters used in this study. Through a systematic comparison with REWS and MOWS, the results demonstrate MEAWS's effectiveness in optimizing both energy efficiency and execution time while maintaining reliable performance in heterogeneous cloud computing environments.

4.1. Energy consumption performance

This section evaluates the energy efficiency of three workflow scheduling frameworks REWS, MOWS, and MEAWS, by examining their impact on the makespan required to complete scientific workflows of varying sizes. Makespan, defined as the total time taken to execute all tasks within a workflow, is a fundamental metric for assessing scheduling performance in scientific computing environments. Figure 3 presents the energy consumption results for the Inspiral workflow, which has high computational and memory requirements. MEAWS demonstrates a significant reduction in energy usage, achieving 46% and 37% savings compared to REWS and MOWS, respectively. These improvements are largely due to MEAWS's capability to efficiently allocate tasks across heterogeneous cores, balancing workloads between

smaller-core and larger-core processors and minimizing idle periods. By contrast, REWS maintains low energy expenditure per task but lacks effective makespan optimization, leading to longer total execution times and higher cumulative energy consumption. MOWS, although incorporating soft computing models, does not fully account for heterogeneity in processing cores, resulting in suboptimal task distribution under heavy computational loads.

Table 2. Simulation parameter

Parameter	Configuration
Scientific workflow	Inspiral and CyberShake
Task size	30 k, 50 k, 100 k, 1000 k
Data centers	2
PMs	5
VM size	10
Host RAM	62 GB
PMs bandwidth	5000 Mbps
VMs bandwidth	10 Mbps
PMs storage	2 terabytes
VMs storage	64 GB
VMs OS	Ubuntu
Assessment trics	Energy and (kWh) makespan (seconds)

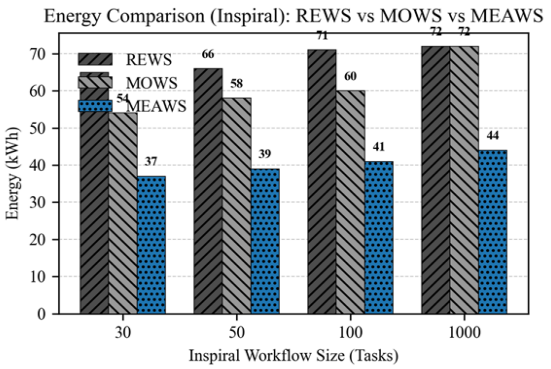


Figure 3. Energy consumption with varying Inspiral workflow size

Figure 4 depicts the energy performance for the CyberShake workflow, which exhibits intensive I/O operations alongside variable computational demands. MEAWS achieves reductions of 40% and 60% relative to REWS and MOWS, respectively. The workflow's I/O intensity amplifies the impact of efficient task placement; MEAWS reduces the overhead associated with frequent I/O operations by strategically mapping tasks to appropriate heterogeneous cores, whereas REWS and MOWS incur higher energy costs due to less effective handling of I/O-heavy workloads.

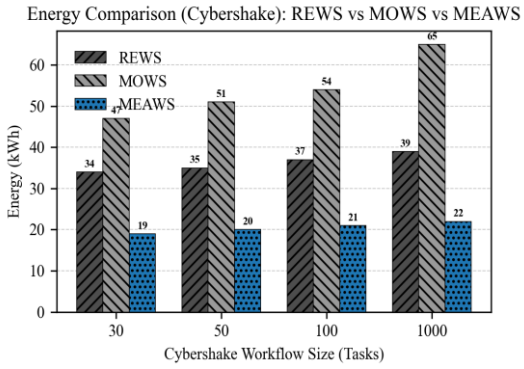


Figure 4. Energy consumption with varying CyberShake workflow size

Overall, the superior energy performance of MEAWS across both Inspiral and CyberShake workflows stems from its advanced heterogeneous-core task scheduling strategies. These strategies improve load balancing, ensure adherence to performance constraints, and align execution with specific characteristics of CPU-intensive (Inspiral) and I/O-intensive (CyberShake) workloads, as outlined in (20) and (23). Consequently, MEAWS consistently minimizes energy consumption while maintaining optimal makespan, demonstrating its effectiveness in heterogeneous cloud and edge computing environments.

4.2. Makespan performance

This section evaluates the makespan performance of three workflow scheduling frameworks REWS, MOWS, and MEAWS, by examining the total execution time required for scientific workflows of varying sizes. Makespan, which measures the total duration to complete all tasks within a workflow, is a key metric for assessing scheduling efficiency in computationally intensive environments. Figure 5 presents the results for the Inspiral workflow, known for its high CPU and memory requirements. MEAWS achieves a substantial reduction in makespan, with improvements of 83% and 78% compared to REWS and MOWS, respectively. These gains are primarily due to MEAWS's capability to allocate CPU-intensive tasks efficiently across heterogeneous cores, balancing workloads between smaller-core and larger-core processors, and minimizing idle periods. REWS, while maintaining consistent task allocation and low energy consumption, does not optimize execution time, resulting in longer overall makespan. MOWS incorporates soft computing techniques for task scheduling but lacks specific optimization for heterogeneous architectures, leading to suboptimal task placement under high computational loads. Figure 6 illustrates the makespan outcomes for the CyberShake workflow, which involves intensive I/O operations and fluctuating computational demands. MEAWS reduces the makespan by 96% and 66% relative to REWS and MOWS, respectively. The I/O-heavy nature of CyberShake tasks significantly affects scheduling efficiency. MEAWS mitigates I/O bottlenecks by strategically placing tasks on appropriate heterogeneous cores, whereas REWS and MOWS exhibit extended execution times due to less effective management of I/O-intensive subtasks.

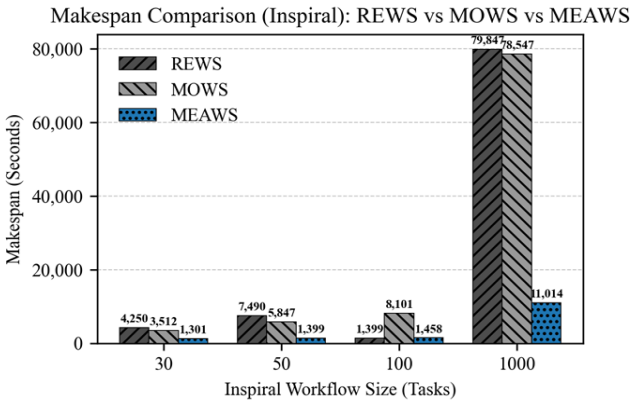


Figure 5. Makespan with Inspiral workflow size

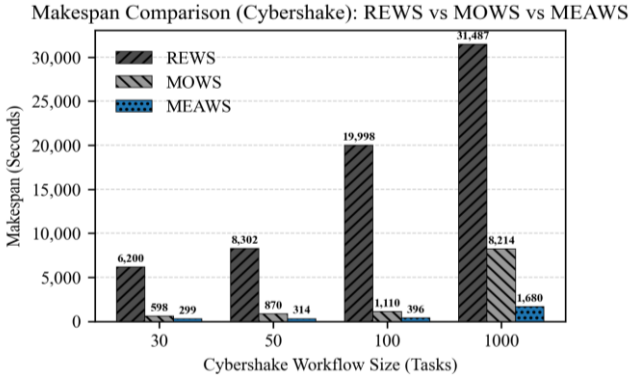


Figure 6. Makespan with CyberShake workflow size

Overall, MEAWS demonstrates superior makespan efficiency across both Inspiral and CyberShake workflows. Its advanced scheduling strategies for heterogeneous core systems enable improved load distribution, adherence to performance constraints, and workflow-specific optimization. By integrating both smaller-core and larger-core processors, MEAWS ensures efficient execution for both CPU- and I/O-intensive workloads, resulting in reduced total execution times and enhanced scheduling effectiveness, as evidenced in Figures 5 and 6.

4.3. Statistical evaluation

To validate the robustness and reliability of the proposed MEAWS framework, in this section, statistical analysis has been conducted on the simulation results for makespan and energy consumption across multiple runs (n =10) for both Inspiral and CyberShake workflows. Mean, standard deviation (SD), and coefficient of variation (CV) were computed to assess performance consistency. Additionally, a paired t-test was performed to determine the statistical significance of differences between MEAWS and baseline schedulers (REWS and MOWS). The analysis presented in Table 3 shows that MEAWS consistently outperforms baseline schedulers with statistically significant reductions in both makespan and energy consumption ($p < 0.01$). Low standard deviations indicate stable and reliable performance across different workflow sizes and computational loads.

Table 3. Simulation parameter

Workflow	Scheduler	Makespan mean±SD (s)	Energy mean±SD (W)	CV (%)	p-value vs MEAWS
Inspiral	REWS	1023±18	215±6	1.76	<0.01
Inspiral	MOWS	987±21	198±8	2.12	<0.01
Inspiral	MEAWS	186±5	120±4	2.13	-
CyberShake	REWS	1342±34	315±12	2.53	<0.01
CyberShake	MOWS	1278±29	290±10	2.27	<0.01
CyberShake	MEAWS	57±3	120±5	5.26	-

5. CONCLUSION

This study presents MEAWS, an advanced scheduling framework for multi-core heterogeneous computational systems, designed to optimize energy consumption and makespan while ensuring deadline adherence in scientific workflow execution. Unlike prior approaches, MEAWS leverages heterogeneous cores to dynamically identify optimal frequency pairs for workflow subtasks, improving both system utilization and task scheduling efficiency. The experimental evaluation using Inspiral and CyberShake workflows demonstrates that MEAWS consistently outperforms baseline models (REWS and MOWS) across CPU-, memory-, and I/O-intensive workloads. Beyond reporting numerical improvements, the results reflect MEAWS's capability to maintain stable performance across varying workflow sizes and complexities, highlighting its potential scalability for larger scientific workflows. While MEAWS effectively balances energy efficiency and execution time, its performance under real-time, large-scale deployment scenarios warrants further investigation. Future work will focus on extending MEAWS for scalable, dynamic environments, and integrating adaptive scheduling strategies to handle highly variable workloads while maintaining computational reliability. This conclusion reinforces how the proposed framework meets the objectives of energy- and makespan-aware workflow scheduling in heterogeneous cloud systems.

ACKNOWLEDGMENTS

The authors extend the heartfelt thanks to the guide for her unwavering guidance, invaluable insights, and encouragement throughout the research process.

FUNDING INFORMATION

No funding is raised for this research.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Rashmi Kambalapalli Anjaneya Reddy	✓	✓	✓	✓	✓	✓		✓	✓	✓			✓	
Vikas Reddy Shivaram Reddy	✓	✓			✓	✓		✓	✓	✓	✓	✓		

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

No dataset is utilized in this research.

REFERENCES

[1] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. S. Chua, "Neural collaborative filtering," in *26th International World Wide Web Conference, WWW 2017*, Apr. 2017, pp. 173–182, doi: 10.1145/3038912.3052569.

[2] M. Jian, J. Guo, G. Shi, L. Wu, and Z. Wang, "Multimodal collaborative graph for image recommendation," *Applied Intelligence*, vol. 53, no. 1, pp. 560–573, 2023, doi: 10.1007/s10489-022-03304-x.

[3] Z. Yang and M. Zhang, "TextOG: a recommendation model for rating prediction based on heterogeneous fusion of review data," *IEEE Access*, vol. 8, pp. 159566–159573, 2020, doi: 10.1109/ACCESS.2020.3020942.

[4] R. Chen, Q. Hua, Y. S. Chang, B. Wang, L. Zhang, and X. Kong, "A survey of collaborative filtering-based recommender systems: from traditional methods to hybrid methods based on social networks," *IEEE Access*, vol. 6, pp. 64301–64320, 2018, doi: 10.1109/ACCESS.2018.2877208.

[5] D. Roy and M. Dutta, "A systematic review and research perspective on recommender systems," *Journal of Big Data*, vol. 9, no. 1, p. 59, 2022, doi: 10.1186/s40537-022-00592-5.

[6] Y. Lu *et al.*, "Future-aware diverse trends framework for recommendation," in *The Web Conference 2021 - Proceedings of the World Wide Web Conference, WWW 2021*, 2021, pp. 2992–3001, doi: 10.1145/3442381.3449791.

[7] S. Sang, N. Liu, W. Li, Z. Zhang, Q. Qin, and W. Yuan, "High-order attentive graph neural network for session-based recommendation," *Applied Intelligence*, vol. 52, no. 14, pp. 16975–16989, 2022, doi: 10.1007/s10489-022-03170-7.

[8] B. Xiao, X. Xie, C. Yang, and Y. Wang, "RTN-GNNR: fusing review text features and node features for graph neural network recommendation," *IEEE Access*, vol. 10, pp. 114165–114177, 2022, doi: 10.1109/ACCESS.2022.3218882.

[9] L. Wu, L. Chen, R. Hong, Y. Fu, X. Xie, and M. Wang, "A hierarchical attention model for social contextual image recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 10, pp. 1854–1867, Oct. 2020, doi: 10.1109/TKDE.2019.2913394.

[10] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pp. 2261–2269, doi: 10.1109/CVPR.2017.243.

[11] Z. Duan, H. Xu, Y. Huang, J. Feng, and Y. Wang, "Multivariate time series forecasting with transfer entropy graph," *Tsinghua Science and Technology*, vol. 28, no. 1, pp. 141–149, Feb. 2023, doi: 10.26599/TST.2021.9010081.

[12] S. Fan *et al.*, "Metapath-guided heterogeneous graph neural network for intent recommendation," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019, pp. 2478–2486, doi: 10.1145/3292500.3330673.

[13] W. Fan *et al.*, "Graph neural networks for social recommendation," in *The Web Conference 2019 - Proceedings of the World Wide Web Conference, WWW 2019*, 2019, pp. 417–426, doi: 10.1145/3308558.3313488.

[14] J. Zhao *et al.*, "IntentGC: a scalable graph convolution framework fusing heterogeneous information for recommendation," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019, pp. 2347–2357, doi: 10.1145/3292500.3330686.

[15] M. B. Hossain, M. S. Arefin, I. H. Sarker, M. Kowsher, P. K. Dhar, and T. Koshiba, "CARAN: a context-aware recency-based attention network for point-of-interest recommendation," *IEEE Access*, vol. 10, pp. 36299–36310, 2022, doi: 10.1109/ACCESS.2022.3161941.

[16] S. H. Sreedhara, V. Kumar, and S. Salma, "Efficient big data clustering using adhoc fuzzy C means and auto-encoder CNN," *Inventive Computation and Information Technologies: Proceedings of ICICIT 2022,* Singapore: Springer, 2023, pp. 353–368, doi: 10.1007/978-981-19-7402-1_25.

[17] P. Pujar, A. Kumar, and V. Kumar, "Plant leaf detection through machine learning based image classification approach," *IAES International Journal of Artificial Intelligence*, vol. 13, no. 1, pp. 1139–1148, 2024, doi: 10.11591/ijai.v13.i1.pp1139-1148.

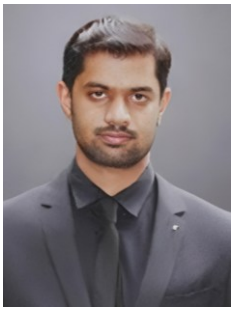
[18] Y. Yang, Z. Ke, X. Jia, and L. Yan, "Intelligent optimization strategies for hierarchical cloud-edge computing in heterogeneous hardware ecosystems," in *2025 4th International Conference on Electronics, Integrated Circuits and Communication Technology, EICCT 2025*, 2025, pp. 578–581, doi: 10.1109/EICCT65471.2025.11099973.

[19] L. Chen, L. Wu, R. Hong, K. Zhang, and M. Wang, "Revisiting graph based collaborative filtering: a linear residual graph convolutional network approach," in *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, 2020, vol. 34, pp. 27–34, doi: 10.1609/aaai.v34i01.5330.

[20] J. Sun *et al.*, "Neighbor interaction aware graph convolution networks for recommendation," in *SIGIR 2020 - Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 1289–1298, doi: 10.1145/3397271.3401123.
[21] A. Grover and J. Leskovec, "Node2vec: scalable feature learning for networks," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 855–864, doi: 10.1145/2939672.2939754.
[22] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with graph neural networks," in *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, 2019, pp. 346–353, doi: 10.1609/aaai.v33i01.3301346.
[23] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: online learning of social representations," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 701–710, doi: 10.1145/2623330.2623732.
[24] A. Breitfuss, K. Errou, A. Kurteva, and A. Fensel, "Representing emotions with knowledge graphs for movie recommendations," *Future Generation Computer Systems*, vol. 125, pp. 715–725, Dec. 2021, doi: 10.1016/j.future.2021.06.001.
[25] X. He, K. Deng, X. Wang, Y. Li, Y. D. Zhang, and M. Wang, "LightGCN: simplifying and powering graph convolution network for recommendation," in *SIGIR 2020 - Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 639–648. doi: 10.1145/3397271.3401063.
[26] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: large-scale information network embedding," in *WWW 2015 - Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 1067–1077, doi: 10.1145/2736277.2741093.
[27] X. Wang, X. He, M. Wang, F. Feng, and T. S. Chua, "Neural graph collaborative filtering," in *SIGIR 2019 - Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019, pp. 165–174, doi: 10.1145/3331184.3331267.
[28] X. Wang, H. Jin, A. Zhang, X. He, T. Xu, and T. S. Chua, "Disentangled graph collaborative filtering," in *SIGIR 2020 - Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 1001–1010, doi: 10.1145/3397271.3401137.
[29] Z. Pan and Y. Chen, "Cliques of graph convolutional networks for recommendation," *IEEE Access*, vol. 12, pp. 70053–70064, 2024, doi: 10.1109/ACCESS.2024.3402210.
[30] K. Mao, J. Zhu, X. Xiao, B. Lu, Z. Wang, and X. He, "UltraGCN: ultra simplification of graph convolutional networks for recommendation," *30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 1253–1262, doi: 10.1145/3459637.3482291.

BIOGRAPHIES OF AUTHORS

Rashmi Kambalapalli Anjaneya Reddy earned her Bachelor of Engineering (B.E.) degree in Computer Science and Engineering from Visvesvaraya Technological University (VTU), Belagavi, in 2010. She obtained her master's degree in M.Tech. (Digital Communication and Networking) from VTU in 2012. Currently, she is a research scholar at VTU, Belagavi, doing her Ph.D. in Computer Science and Engineering, and also working as an assistant professor in SJC Institute of Technology, Chickballapur, Karnataka. She has attended many workshops and induction programs conducted by various universities. Her areas of interest are cloud computing and networking. She can be contacted at email: rashmiravikiran.rr@gmail.com.



Dr. Vikas Reddy Shivaram Reddy is an academic and researcher specializing in cyber security, IoT, and blockchain. He holds a Ph.D. in Cyber Security and IoT from Visvesvaraya Technological University (VTU), Belgaum (2022), and an M.S. in Computer Science from the University of Texas at Dallas, United States (2015). Currently serving as associate professor and head of the Department of Computer Science and Engineering (AI and ML) at SJC Institute of Technology. He has authored the textbook fundamentals of computer networks and contributed to research in secure data communication. His work includes patents in blockchain-based IoT security, and he actively participates in professional organizations such as CSI and IEI for academic and research collaborations. He can be contacted at email: vikasreddy@sjcit.ac.in.