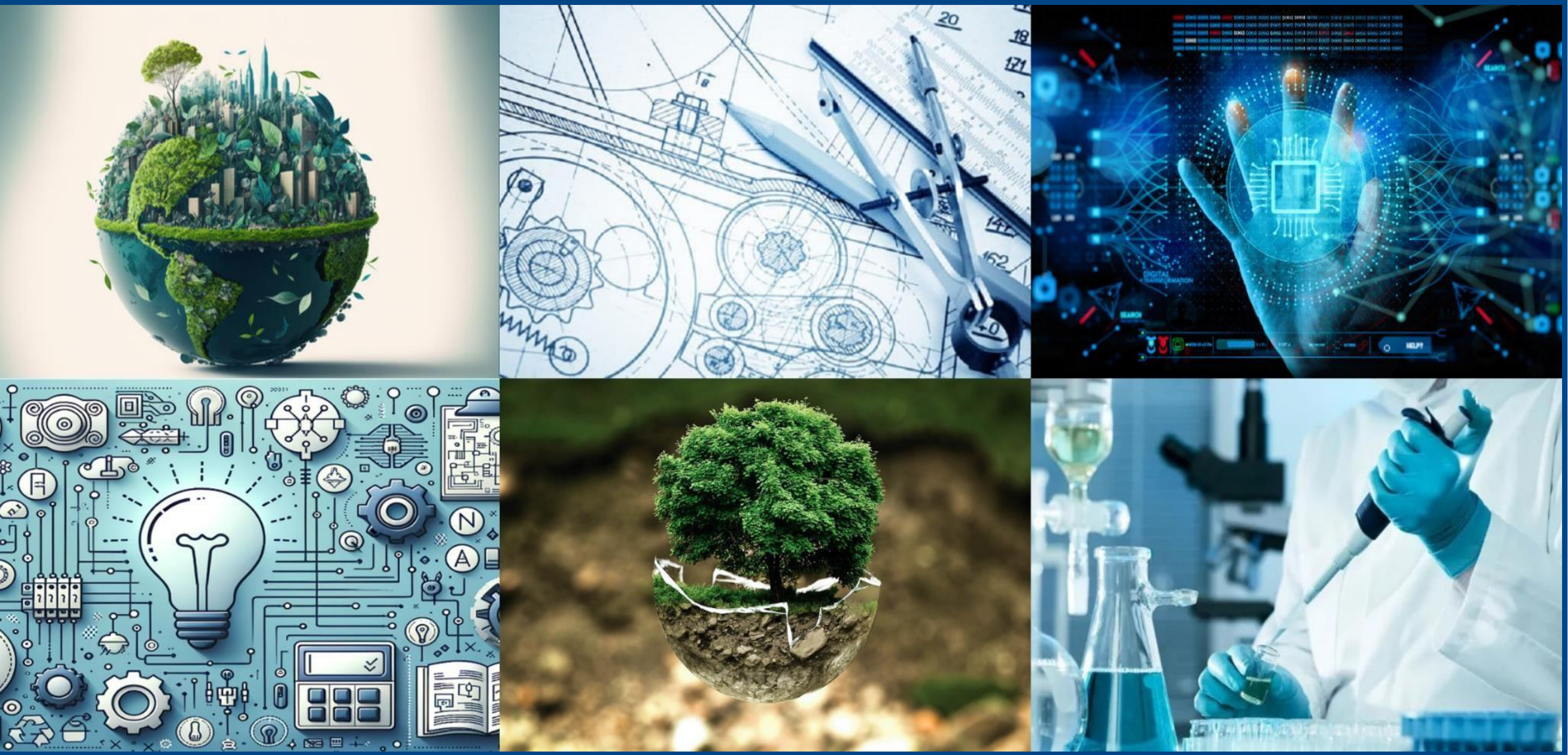
ISSN: 2582-7219

International Journal of Multidisciplinary Research in Science, Engineering and Technology

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.206

Volume 8, Issue 8, August 2025

© 2025 IJMRSET | Volume 8, Issue 8, August 2025| DOI:10.15680/IJMRSET.2025.0808119

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

A NOVEL EVALUATION METHOD ON SOFTWARE DESIGN REUSABILITY FOR BETTER SOFTWARE MAINTENANCE

Praveen K S, Dr.L. Manjunath Rao

Associate Professor, Department of MCA, East West Institute of Technology, Bengaluru, India

Professor, Department of MCA, Dr. Ambedhkar Institute of Technology, Bengaluru, India

ABSTRACT: Scientific workflows typically represented as Directed Acyclic Graphs (DAGs) have become indispensable for orchestrating complex, data-intensive computational experiments in scientific and industrial domains . Each node in a DAG denotes a distinct computational or data operation (e.g., data collection, storage, aggregation, analysis, or simulation), and the directed edges capture precedence and data dependencies among these tasks. Efficient scheduling of these tasks is critical to guarantee that interdependent operations complete in the correct order while making judicious use of available resources. As workflow sizes and complexities escalate, traditional on-premises infrastructures struggle to satisfy the dynamic computational demands, prompting a migration toward cloud-based environments .

I. INTRODUCTION

Cloud platforms offer on-demand, scalable provisioning of compute and storage resources, which aligns well with the elastic requirements of scientific workflows [5]. However, centralized cloud data centers can suffer from resource contention, queuing delays, and significant energy consumption issues that become more pronounced under peak workloads or with exceptionally large-scale workflows [6]. To mitigate such constraints, multi-cloud strategies have been explored, distributing workload across several cloud providers to enhance resource availability and fault tolerance [7]. Yet, inter-cloud communication overheads and latency variability often undermine performance gains in real-world scenarios [8].

Edge computing has emerged as a complementary paradigm by shifting computation closer to data sources, thereby reducing end-to-end latency and enabling near-real-time processing [9]. When combined with heterogeneous multi-core architectures comprising, for instance, high-performance “big” cores alongside energy-efficient “little” cores edge-cloud hybrid infrastructures can further accelerate workflow execution and curb energy expenditure [10], [11]. Despite these advantages, existing scheduling schemes seldom integrate cost considerations (i.e., both time and energy) with the strategic reutilization of idle multi-core resources across edge and cloud tiers. Moreover, most multi-objective approaches focus on homogeneous clouds or static resource assignment, leaving a gap in methods that dynamically exploit heterogeneous cores to meet tight deadlines without inflating energy costs [12], [13].

A key research challenge, therefore, is to devise a scheduling framework that not only respects the DAG’s precedence constraints and deadline requirements but also minimizes energy and time overhead by reallocating underutilized multi-core resources across the edge-cloud continuum. While prior work has addressed makespan–cost trade-offs [14], [15], these solutions typically overlook the potential of recycling idle computational slots on heterogeneous cores, especially within resource-constrained edge nodes. In the absence of adaptive mechanisms to identify optimal frequency–core pairings for each task balancing energy consumption and execution delay, deadline violations and energy wastage remain unavoidable.

To bridge these gaps, we propose a novel Multi-Agent Deep Reinforcement Learning (MADRL) framework, designated as Cost-effective Resource Reutilization for Scientific Workflow Applications (CRRU-MADRL). CRRU-MADRL jointly optimizes makespan and energy metrics by dynamically selecting frequency states for heterogeneous cores and reutilizing idle resources in both edge and cloud servers. Each agent in the framework corresponds to a sub-workflow or task cluster, learning to negotiate resource allocations and timing decisions that minimize idle periods while satisfying precedence and deadline constraints. By coordinating multiple agents through a shared reward structurepenalizing energy-intensive configurations and missed deadlines, CRRU-MADRL continuously adapts to varying workflow demands and resource availabilities, thereby reducing overall cost (time + energy) and improving resource utilization in edge-cloud platforms.

II. LITERATURE SURVEY

In recent years, cloud computing has become a pivotal platform for executing complex workflows, necessitating efficient scheduling algorithms that balance performance metrics such as execution time, cost, energy consumption, and reliability. This literature survey examines several notable studies that propose innovative approaches to workflow scheduling in cloud environments, focusing on their methodologies, optimized metrics, performance improvements, significance, limitations, and future research directions. M. Fan et al [16], introduced a scheduling algorithm that dynamically adjusts task priorities and optimizes critical tasks within budget constraints. The approach involves analyzing task dependencies and resource requirements to allocate resources effectively. The primary metrics optimized are execution cost and makespan, ensuring that workflows are completed within the specified budget and time constraints. Their method demonstrates a significant reduction in execution costs and makespan compared to traditional scheduling algorithms, highlighting its efficiency in resource utilization. By focusing on critical tasks and adjusting priorities dynamically, their approach enhances the efficiency and cost-effectiveness of workflow executions in cloud environments. However, the algorithm may face challenges in highly dynamic cloud environments where resource availability fluctuates rapidly, potentially affecting its adaptability. S. Tao et al. [17], presents the deadline-budget-aware algorithm, which applies Ant Colony Optimization (ACO) to schedule workflows under strict deadline and budget constraints. The algorithm models the scheduling problem as a graph and utilizes artificial ants to explore optimal paths that minimize execution costs while meeting deadlines. The focus is on minimizing execution cost without violating predefined deadlines and budget limits. Experimental results indicate that their method outperforms several state-of-the-art methods, particularly in complex workflows like CyberShake, by achieving lower costs and better adherence to deadlines. The integration of ACO provides a robust framework for handling the combinatorial nature of workflow scheduling, offering a balance between exploration and exploitation in search of optimal solutions. However, the performance of DB-ACO can be sensitive to the parameter settings of the ACO algorithm, which may require fine-tuning for different workflow scenarios.

S. -E. Chafi et al. [18], et al. proposes a Particle Swarm Optimization (PSO)-aware model tailored for scheduling workflows in heterogeneous fog computational platforms. The algorithm aims to optimize both execution time and energy consumption by considering the unique characteristics of fog nodes. The primary metrics optimized are makespan and energy consumption, addressing both performance and sustainability concerns. The PSO-aware model achieves a notable reduction in makespan and energy usage compared to baseline scheduling approaches, demonstrating its effectiveness in fog computing contexts. By optimizing for energy efficiency alongside execution time, the study contributes to the development of sustainable computing practices in emerging fog environments. However, the algorithm's performance may be influenced by the heterogeneity of the fog nodes and the variability in network conditions, which could affect its generalizability. J. Perez-Valero et al. [19], study introduces an Network Functions Virtualization (NFV) concept to support energy-aware adaptive scaling approach. The method dynamically adjusts the number of active servers based on workload demands and reliability requirements. The focus is on minimizing energy consumption while maintaining the reliability of NFV services. The adaptive scaling approach results in significant energy savings without compromising service reliability, as evidenced by experimental evaluations. The research addresses the critical balance between energy efficiency and reliability in NFV infrastructures, contributing to more sustainable and dependable cloud services. The approach may require accurate workload prediction models to effectively scale resources, which can be challenging in unpredictable traffic scenarios.

S. Ijaz et al. [20], presents an improved multi-objective differential evolution algorithm designed to schedule tasks in fog computing environments. The algorithm simultaneously optimizes energy consumption, execution time, and cost, considering the constraints inherent in fog computing. The key metrics optimized include energy efficiency, makespan, and execution cost, providing a comprehensive approach to resource management. The proposed algorithm demonstrates superior performance in balancing the trade-offs between energy consumption, time, and cost compared to existing scheduling methods. By addressing multiple objectives, the study offers a holistic solution for task scheduling in fog computing, enhancing both efficiency and cost-effectiveness. The complexity of the algorithm may lead to increased computational overhead, which could be a concern in resource-constrained fog environments. X. Wang et al. [21], employs coalition reinforcement learning (CRL) to optimize multi-workflow scheduling and cloud bundle provisioning. The approach forms adaptive coalitions among workflows for efficient resource utilization. The metrics optimized are cost, execution time, and resource utilization. The model achieves better cost efficiency and faster execution times compared to traditional heuristic-based approaches. The model enhances workload management in large-scale cloud systems by dynamically adjusting resource allocations. However, the model induces high computational complexity in large-scale workflow scenarios.

H. Lahza et al. [22], a deep reinforcement learning (DRL) approach is proposed to optimize resource allocation across edge and cloud layers for workflow scheduling. The metric optimized are energy consumption, execution time, and resource utilization. The model achieves lower energy consumption while maintaining workflow execution efficiency compared to conventional methods. Addresses the challenge of balancing workload distribution between cloud and edge resources. However, The model's efficiency is affected by network fluctuations and edge resource constraints. Krishna et al. [23], utilizes the Asynchronous Advantage Actor-Critic (A3C) reinforcement learning model to prioritize workflow scheduling in cloud environments. The metric optimized: Makespan, energy consumption, and cost. The model shows significant improvements in reducing makespan and cost compared to conventional scheduling methods. The approach effectively handles dynamic workload variations by leveraging reinforcement learning techniques. The training phase is computationally expensive, making it less suitable for real-time scheduling.

Mangalampalli et al. [24], presented a deep reinforcement learning-based scheduler that considers energy efficiency and temperature constraints in cloud workflow scheduling. The metric optimized are energy consumption, execution time, and thermal efficiency. The model demonstrates reduced energy consumption and improves thermal stability in cloud data centers. The model ensures sustainable and efficient workflow execution by integrating energy and thermal considerations. However, the system's complexity increases due to additional thermal monitoring requirements. L. Zhang et al. [25], Methodology: Proposes reliability-aware scheduling strategies considering energy constraints in cloud computing. The metric optimized are reliability, energy efficiency, and execution time. The model enhances system reliability while maintaining optimal energy usage compared to conventional models. The approach ensures high workflow reliability without significant trade-offs in energy consumption. However, the model increased computational overhead due to reliability assessments. The Table 1, provide a deep rooted analysis of various scheduling methods.

Author, Ref No.	**Cloud Model Used**	**Metrics Optimized**	**Methodology Introduced**	**Optimization Strategy**	**Workflow Used**	**Result Studied**
M. Fan et al. [16]	Cloud Computing	Budget, Makespan	Priority Adjustment and Critical Task Optimization	Heuristic-based Approach	Scientific workflow	Improved Budget Utilization and Task Scheduling Efficiency
S. Ijaz et al. [20]	Fog Computing	Energy, Time, Cost	Improved Multi-Objective Differential Evolution (IMODE)	Evolutionary Algorithm	Scientific workflow	Enhanced Energy Efficiency and Cost Reduction
H. Lahza et al. [22]	Edge-Cloud	Resource Utilization, Latency, Cost	Adaptive Multi-Objective Resource Allocation using Deep Reinforcement Learning (DRL)	Deep Reinforcement Learning	Scientific workflow	Improved Resource Utilization and Reduced Latency
M.S.R. Krishna et al. [23]	Cloud Computing	Makespan, Task	Prioritized Workflow	Deep Reinforcement	Scientific workflow	Enhanced Task

		Prioritization	Scheduler Using Asynchronous Advantage Actor-Critic (A3C) Algorithm	Learning		Scheduling and Reduced Execution Time
S. Sudheer Mangalampalli et al. [24]	Cloud Computing	Energy, Temperature, Workflow Efficiency	Energy and Temperature Aware Deep Reinforcement Learning Workflow Scheduler	Deep Reinforcement Learning	Scientific workflow	Reduced Energy Consumption and Maintained Temperature Constraints
L. Zhang et al. [25]	Cloud Computing	Reliability, Energy Consumption	Reliability Enhancement Strategies for Workflow Scheduling	Heuristic and Optimization-Based Approach	Scientific workflow	Improved Workflow Reliability and Energy Efficiency

These studies contribute significantly to workflow scheduling in cloud and edge computing by leveraging advanced optimization techniques such as reinforcement learning and heuristic models. The reviewed literature focuses on workflow scheduling in cloud and edge computing environments, optimizing various performance metrics through advanced methodologies. Several studies employ Deep Reinforcement Learning (DRL) [22], [23], [24], to enhance resource allocation, task prioritization, and energy efficiency. Others utilize heuristic-based [16], [24] and evolutionary optimization techniques [20], to balance constraints like budget, reliability, and execution time. The primary objectives across these works include reducing makespan, cost, energy consumption, and latency, while improving task scheduling efficiency, reliability, and resource utilization. Overall, these approaches contribute significantly to efficient cloud workflow management by integrating adaptive and intelligent scheduling strategies. However, challenges remain in computational complexity, network variability, and real-time adaptability. Thus, considering research challenges in next section introduces an efficient scheduling mechanism that consider maximizing resource usage for reducing both energy and makespan.

III. OBJECTIVES OF THE RESEARCH WORK

Theresearch objectives are given below.

1. To design makespan-energy and cost-effective metrics for virtual resource function optimization for scientific workflow provisioning in edge-cloud platforms.
2. To design Multi-agent Reinforcement Learning Framework for cost effective scientific workflow application resource reutilization in edge-cloud platforms.
3. To develop service function chain reutilization optimization for meeting scientific workflow task multi-level deadline constraint.
4. To design and develop cost-reliability metrics for reusability of virtual resource function and service chaining optimization for execution complex multiple scientific workflows in edge-cloud platform.

Multi-agent deep reinforcement learning framework for cost effective scientific workflow application resource reutilization in Edge-Cloud platform

This section presents a cost-effective scientific workflow application resource reutilization aware scheduler designed using MADRL. The architecture of CRRU-MADRL in edge-cloud platforms given in Figure 1.

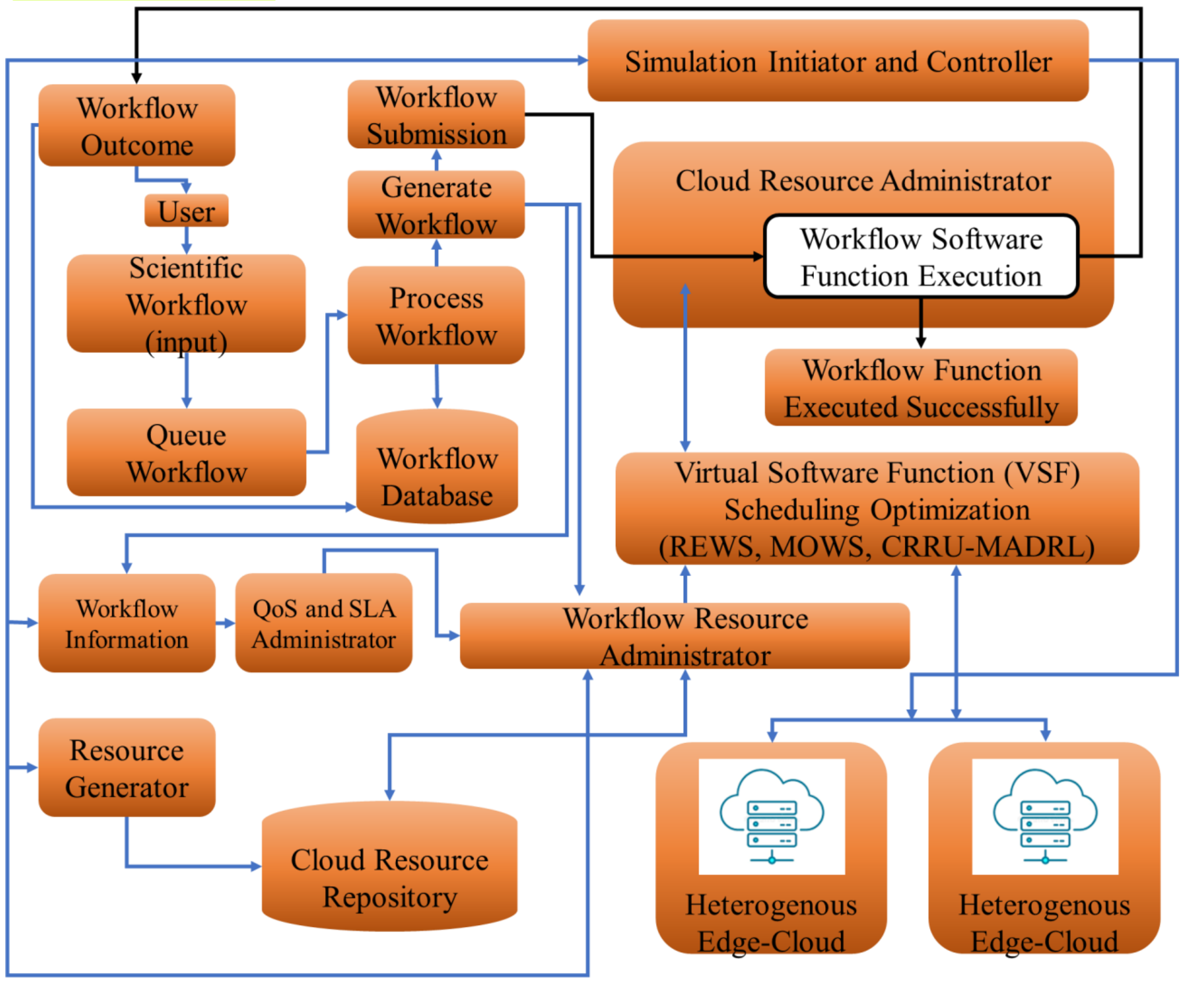

Simulation Initiator and Controller
Workflow Outcome
Workflow Submission
User
Generate Workflow
Scientific Workflow (input)
Process Workflow
Queue Workflow
Workflow Database
Cloud Resource Administrator
Workflow Software Function Execution
Workflow Function Executed Successfully
Virtual Software Function (VSF) Scheduling Optimization (REWS, MOWS, CRRU-MADRL)
Workflow Information
QoS and SLA Administrator
Workflow Resource Administrator
Resource Generator
Cloud Resource Repository
Heterogenous Edge-Cloud
Heterogenous Edge-Cloud

Figure 1. Proposed Heterogenous Cloud Computing Architecture for Execution of Scientific Workflows.

IV. SYSTEM MODEL

Depending on the user's specified QoS/SLA requirements, the CRA invokes the appropriate scheduling strategy: **REWS or MOWS** for traditional execution. CRRU-MADRL model for cost-effective scheduling ensuring lesser energy, time, with enhanced resource utilization in heterogeneous edge-cloud settings. The CRA then queries the Workflow Resource Administrator (WRA) to assess the current resource availability across edge and cloud servers. This includes evaluating: Heterogeneous multi-core processors, available memory, bandwidth, network latency and node energy status. The CRRU-MADRL framework, employs multiple reinforcement learning agents each responsible for a cluster of tasks or a DAG level. The agents operate cooperatively to: Dynamically select optimal frequency–core pairings using DVFS (Dynamic Voltage and Frequency Scaling). Reutilize idle resources (underutilized CPU cores, memory buffers, transmission paths) across both edge and cloud layers. Balance trade-offs between makespan, energy consumption, and resource utilization. Learn and adapt scheduling policies using feedback from task execution results. To enhance the flexibility and efficiency of scientific workflow execution, CRRU-MADRL integrates Virtual Software Functions (VSFs) that represent modular service functions like: Aggregation Function: Merges intermediate results from distributed sources at edge nodes to reduce redundant data transfer to cloud. Pipelining Function: Enables sequential chaining of dependent tasks across nodes with overlapping compute–transfer stages for speedup. Computation Function: Executes task logic on appropriate compute units (e.g., AI models, simulations, analytics). Transmission Function: Handles data migration between edge-edge or edge-cloud depending on network bandwidth and latency. These VSFs are abstracted as reusable execution units. The CRRU-MADRL model dynamically decides when and where to deploy, reuse, or migrate these functions based on: Task characteristics (data volume, urgency, compute complexity), Network conditions (bandwidth, latency), and Resource state (CPU load, energy budget). Reutilization is achieved through: Function Caching: Previously instantiated Virtual Service Functions (VSFs) are cached at edge or cloud nodes and reused for recurring tasks. This caching mechanism reduces startup latency and avoids the overhead of redeploying identical functions for frequently appearing sub-tasks across multiple workflow executions. By retaining service state and configuration at local nodes, CRRU-MADRL minimizes data movement and accelerates execution, especially for repetitive scientific workflow components such as preprocessing, feature extraction, or simulation kernels.

Cost and Resource Reutilization Model

Directed Acyclic Graphs (DAG) are used for modeling the workflows that are scheduled by the proposed hierarchical scheduling framework. The tasks of a workflow are represented by the set of nodes, $V = \{v_0, v_1 \ldots$ and the precedence constraints between tasks are represented by the set of edges, $E = \{(v_i, v_j) | v_i, v_j \in$ of a DAG, $G = (V,$. It is also assumed that workflows are containerized (each task is a container). Workflows are to be scheduled across a federation of geo-distributed datacenters $DC = \{dc_1, dc_2, .. d$ which are powered through a combination of green energy from renewable energy sources and brown-energy from the grid. Therefore, at any instance, the total power consumption, P_{tc} of the datacenter federation is the sum of green power, P_{gr} and brown power, P_{brc} consumed by the underlying cloud infrastructures and operations. The datacenter, comprises of a set of heterogeneous severs, $\{m_1, m_2, .. m$. Power consumed by a server, is calculated using the CPU utilization-based power model presented in [31] as follows:

$$P_{m_i} = \begin{cases} P_{m_i}^{idle} + \left(P_{m_i}^{dynamic} - P_{m_i}^{idle}\right) \cdot u_{m_i}, & if\, u_{m_i} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where P_{m}^{i} is the idle power consumption of the server which is a constant regardless of its current utilization and $P_{m_i}^{dyna}$ is the dynamic power consumption which is dependent on the current server utilization, ι. Accordingly, the total power consumed by the datacenter, during the th time interval is computed as follows:

$$P_i^{Total}(k) = \sum_{i=1}^{\lambda_i} P_{m_i}(k) \tag{2}$$

The objective of the proposed hierarchical scheduling framework is to minimize total brown energy utilization of the cloud datacenter federation while also optimizing workflow execution time. The total brown energy consumption at the datacenter during the *k*th time interval can be represented as:

$$P_i^{brown}(k) = P_i^{Total}(k) - P_i^{green}(k) \tag{3}$$

where P_i^{green} is the total green energy available at the datacenter during the *k*th time interval. Accordingly, the primary objective of the global scheduler which mainly focuses on minimizing the brown energy usage during the th time interval can be represented as follows:

$$\text{Min: } P_{total}^{brown}(k) = \sum_{i=1}^{N} P_{m_i}(k) \tag{4}$$

Once a task is assigned to a datacenter , the local scheduler allocates the task to a server that jointly minimizes the total energy consumption and execution time of the task. Hence, the objective of the local scheduler during the *k*th interval can be represented as follows:

$$\text{Minmize: } \sum_{j=1}^{N} \alpha T_{t_j} + (1-\alpha) E_{t_j} \tag{5}$$

where is the total number of tasks executed during the *k*th time interval. and denotes the total execution time and total energy consumption associated with the execution of task , respectively. The execution time includes the maximum data transfer time from predecessor nodes, computation time of the task as well as the waiting time of the task at the node (W) before the task is actually executed. It can be computed as follows:

$$T_{t_j} = \frac{L_{t_j}}{F} + WT_{t_j} + \max_{t_i \in pred(t_j)} DT_{t_i,t_j} \tag{6}$$

where, is the size of the task , and is the processing rate of the node to which task is assigned. The ratio is the computation time of the task. DT_t is the data transfer time from the node in which predecessor executed to the execution node of task . If tasks are in the same datacenter, then DT_t can be expressed as the ratio $\frac{DT}{B}$ where DT_t is the size of data to be transferred from to and is the network bandwidth within the same datacenter. If data transfer is between nodes in different datacenters, DT_t can be represented as $\frac{DT}{B_c}$, where B represents the network bandwidth between datacenters. Total energy consumed during the execution of task is computed as follows:

$$EE_{t_j} = T_{t_j} \times [U \times P_{active} + P_{idle}] \tag{7}$$

where is the current CPU utilization level of the execution node, and the rates of power consumption at active and idle states of the processors are denoted by P_{act} and P_i, respectively. Considering the power consumption associated with the transmission of data to be P_{co}, the energy consumed during the transfer of data from predecessor nodes is computed as follows:

$$ET(t_j) = \sum_{t_i \in pred(t_j)} \frac{D_{t_i,t_j}}{B} \times P_{comm} \tag{8}$$

The total energy consumed is the sum of computation and communication energy, and is computed as:

$$E_{t_j} = EE_{t_j} + ET(t_j) \tag{9}$$

The computation time of a task, can be represented as:

$$CT(t_j) = \frac{L_{t_j}}{F} \tag{10}$$

All the precedence constraints of task, must be satisfied before its execution commences. Accordingly, the execution of all the predecessors must be completed, and the output data required for the execution of must be transmitted to the node in which it is scheduled. If is an immediate predecessor of and the size of data to be transferred from to is $D(t_i,$, then the total transmission time ( ) can be denoted as follows:

$$TT(t_i,t_j) = \frac{D(t_i,t_j)}{B} \tag{11}$$

where is the bandwidth between the execution nodes of and . Task execution delay, $TD($ primarily depends on the computation time, $CT($ of the task, and the maximum data transfer time from predecessor nodes, $\max_{t_i \in pred(t_j)} TT(t_i,$. The waiting time, $WT($ before a task gets scheduled also contributes to total execution delay. Accordingly, $TD($ can be represented as:

$$TD(t_j) = CT(t_j) + WT(t_j) + \max_{t_i \in pred(t_j)} TT(t_i,t_j) \tag{12}$$

The finish time, $FT($ of task, that started execution at time, $ST($ can then be expressed as:

$$FT(t_j) = ST(t_j) + TD(t_j) \tag{13}$$

The completion time, of a workflow is equivalent to the time at which that last task of the workflow completes execution. It can be denoted as:

$$MT = \max_{t_i \in pred(t_j)} (FT(t_j)) \tag{14}$$

where represents the set of all tasks of the workflow. The computation cost of that executes in a Node with unit cost per second, can be represented as:

$$CC(t_j) = CT(t_j) * UC \tag{15}$$

The cost of execution, I of a workflow is equivalent to the sum of execution costs of all tasks, and it can be denoted as follows:

$$MC(t_j) = \sum_{t_j \in T} CC(t_j) \tag{16}$$

The objective of the scheduling problem is to minimize the cost of workflow executions, and it can be denoted as follows:

$$\text{Minmize}: \sum_{i=1}^{N} MC_i \tag{17}$$

where is the total number of workflows submitted to the system.In order to achieve cost minimization while ensuring energy efficiency and effective resource reutilization, particularly the reutilization of Virtualized Server Functions (VSFs) we employ a Multi-Agent Deep Reinforcement Learning (MADRL) approach tailored for heterogeneous edge–cloud environments with multi-core processing capabilities.

Multi-Agent Deep Reinforcement Learning Model

Reinforcement Learning (RL), a branch of machine learning, empowers agents to interact with dynamic environments and learn optimal policies through experience-based reward feedback. Each agent observes the current environmental state , selects an action , and transitions to a new state s_t, receiving a corresponding reward . Over time, the agent aims to maximize the cumulative discounted reward:

$$J(\theta) = E_{\pi_\theta}\left[\sum_{t=1}^{\infty} \gamma^t r_t\right] \tag{18}$$

where $\gamma \in (0$ is the discount factor and $\pi_\theta(a_t|$ is the parameterized policy with learnable weights . The policy gradient is given by:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}\left[\nabla_\theta \log\left(\pi_\theta(a|s) Q^{\pi\theta}(s,a)\right)\right] \tag{19}$$

Here, $Q^{\pi\theta}(s,$ is the state–action value function estimating expected returns for executing action in state under policy . To manage complexity and partial observability in distributed edge-cloud environments, a MADRL model is adopted. The system incorporates both **global** and **local** agents, structured hierarchically: a **global scheduler** operates across data centers, while **local schedulers** manage task assignments within individual Physical Machines (PMs). The distributed model is formulated using a **Partially Observable Markov Game (POMG)**, defined as a seven-tuple $(N, S, \phi, \{A_i\}, P, \{O_i\}, \{R$, where denotes agents set, denotes finite states set, denotes starting state distribution, corresponds to action set available to agent , defines state-transition probability function given joint actions $A_1 \times A_2 \times \cdots \times$, represents observation set for agent and $R_i : S \times A_i \rightarrow$ denotes reward function for agent ; where agents interact based on joint action sets and local observations. For agent , the policy gradient with shared critic knowledge is expressed as:

$$\nabla_{\theta_i} J(\theta_i) = E_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_i|O_i) Q^{\pi\theta_i}(x, a_1, a_2, \ldots, a_N)\right] \tag{20}$$

In Eq. (20), $Q^{\pi\theta_i}(x, a_1, a_2, \ldots, c$ denotes critics estimated of state-action value function, taking into account the state and joint actions of all agents. This formulation enables each agent to learn distinct policies while considering joint environmental dynamics and cooperative scheduling behavior.

Global Scheduler

Each workflow consists of containerized tasks subject to Directed Acyclic Graph (DAG) precedence constraints. The global scheduler identifies tasks ready for execution i.e., those whose dependencies have been resolved and invokes the DRL agent accordingly. The **state representation** includes: F: Net green energy surplus or deficit in data center , F: Average processing speed of servers in , U: Current resource utilization level in , and t_j^{c}, t_j^{m}: CPU and memory requirements of task .The **action space** is defined as:

$$A = \{dc_i / dc_i \in \{dc_1, dc_2, \ldots, dc_n\}\} \tag{21}$$

The **reward function** integrates both the environmental sustainability and scheduling efficiency:

© 2025 IJMRSET | Volume 8, Issue 8, August 2025| DOI:10.15680/IJMRSET.2025.0808119

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

$$R_{global} = \left[P^{green}_{dc_i} - P^{total}_{dc_i}\right] + R_{local} \quad (22)$$

Here, $P^{gr}_{dc_i}$ and P^{tc}_{dc} denote the available green energy and total energy demand of data center , respectively, while R_{lc} reflects the performance feedback from the corresponding local scheduler. The training process of actor-critic DRL model for global-scheduler for workflow scheduling is presented in Algorithm 1 as discussed below.

```
Algorithm 1: MADRL model for Global-Scheduler
Input: $\pi_\theta(a$: Actor policy with parameters , $Q_w(s,$: Critic value function with parameters , : Total number of training episodes, : Maximum steps per episode.
Output: Optimized policy for cost-efficient task scheduling.
Step 1.  Start
Step 2.  Initialize actor $\pi_\theta(a$: and critic $Q_w(s,$ with random weights and
Step 3.  For each episode= $1\ t$ do
Step 4.      Reset environment to initial state
Step 5.      For each step= $1\ t$ do
Step 6.          Observe global state : includes { $F$ (green energy surplus/deficit), $F$ (avg. processing speed), $U$ (current utilization), VSF availability, Task profile $t_j^l$, $t_j^m$}
Step 7.          Dispatch task to selected data center via $a_{lc}$
Step 8.          Receive feedback from local-scheduler using Algorithm 2: Local reward $R_{lc}$ and local action $a_{lc}$
Step 9.          Compute global reward: $R_{global} = \left[P^{green}_{dc_i} - P^{total}_{dc_i}\right] + R_{lc}$
Step 10.         Formulate joint action $a_t = (a_{global}, a_{loc}$
Step 11.         Critic evaluates $Q_w(s_t,$
Step 12.         Compute TD error $\delta = R_t + \gamma \cdot Q_w(s_{t+1}, a_{t+1}) - Q_w(s_t,$
Step 13.         Update critic parameters using gradient $\nabla_w($
Step 14.         Update actor parameters using policy gradient: $\nabla_{\theta_i} J(\theta_i) = E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a_i|O_i) Q^{\pi_{\theta_i}}(x, a_1, a_2, \dots, a_l$
Step 15.     End For
Step 16. End For
Step 17. Return optimized policy
Step 18. Stop
```

Local Scheduler and VSF Reutilization

Upon receiving task assignments from the global agent, the local scheduler attempts to allocate them to appropriate PMs based on task requirements and server availability. If resources are unavailable, it alerts the global agent for reassignment. The **local agent's state** includes: PM processing power, Server utilization metrics, Task size and resource demand. The **reward** is calculated in terms of execution time reduction, energy consumption, and **VSF reutilization**. Efficient reuse of previously activated VSFs leads to lower startup costs, reduced migration overhead, and minimal energy waste. The local agent updates its policy to prioritize such reuse while maintaining system-wide QoS goals. Performance feedback is communicated to the global agent to improve future scheduling decisions.

Work carried out so far

This section presents an experimental evaluation of the proposed CRRU-MADRL for executing scientific workflows in a heterogeneous computational system. To assess its effectiveness, CRRU-MADRL is compared against existing scheduling approaches, including Reliable and Efficient Workflow Scheduling (REWS) [25] and Multi-Objective Workflow Scheduling (MOWS) [15], [20]. REWS is selected as a baseline due to its focus on energy minimization while meeting task deadlines, whereas MOPW is designed to optimize resource utilization while reducing makespan and energy consumption. The implementation of CRRU-MADRL, along with baseline models such as REWS and MOWS is carried out using the CloudSim-based simulator [26], [27], [28]. The Inspiral and Cybershake scientific workflow is utilized to assess the scheduler models [29]. Inspiral is characterized by high memory and processing overhead, whereas Cybershake exhibits intensive I/O and processing demands, making them ideal benchmarks for testing scheduler robustness across diverse computational environments. The simulation study evaluates key performance metrics, including makespan and energy consumption. Experiments are conducted on a Windows 11 operating system, running on an Intel Core i7 processor with 16 GB memory with CUDA GPU support having 4 GB memory. The experimental setup consists of five physical hosts and 50 virtual machines, ensuring a realistic cloud execution environment. The simulation parameters considered for this study are detailed in Table 2. By systematically analyzing the performance of CRRU-MADRL against state-of-the-art scheduling models, this study provides a comprehensive evaluation of its capability to optimize energy efficiency and execution time while maintaining computational reliability in heterogeneous cloud computing platforms.

Table 1: Simulation parameter

Parameter	**Configured size**
Scientific workflow	Inspiral and Cybershake
Workflow complexity size	30, 50, 100, 1000
Data centers considered	2
Number of physical machines	10
Number of virtual computing node with smaller-core and larger-core	20
The RAM capacity of the host	32 GB
Bandwidth of Physical machine	1000 Mbps
Bandwidth of Virtual computing node	5 Mbps
Storage size of Physical machine	1 TB
Storage size of virtual computing node	32 GB
virtual computing node operating system	Ubuntu
Hypervisor configuration	Xen
Performance metrics	Energy consumption, makespan

Energy Consumption Performance:

This section assesses the energy consumption of three scheduling frameworks namely REWS, MOWS, and CRRU-MADRL by analyzing the makespan needed to finish execution of scientific workflows of different sizes. The makespan, defined as the total time to execute a given set of tasks, is a critical performance metric in scientific computing. Figure 2 demonstrates the energy consumption performance when executing the Inspiral workflow across varying task sizes. The results show that CRRU-MADRL significantly reduces energy usage by 44.3% and 35.4% when compared to REWS and MOWS, respectively. The results indicate that CRRU-MADRL outperform the baseline models by a considerable margin, making them highly efficient for executing large and complex workflows. Figure 3 exhibits the energy consumption outcome for Cybershake workflows, which include diverse size with varying computational demands. The results reveal that CRRU-MADRL reduces energy consumption by 38.88% and 58.9% compared to REWS and MOWS, respectively. These improvements highlight CRRU-MADRL as superior frameworks in minimizing makespan while processing complex scientific workflows. These improvements highlight CRRU-MADRL as superior frameworks in minimizing energy consumption while processing complex scientific workflows. The extended energy consumption reflected in REWS and MOWS can be credited to the inherent restrictions of each method. REWS primarily focuses on task allocation to VCNs with superior consistency and lower energy spending but does not rank optimizing makespan, resulting in longer execution times contributing increased energies. MOWS, on the other hand, utilizes soft computing models for scheduling tasks but is not specifically tailored for heterogeneous platforms, which leads to suboptimal task scheduling when compared to the advanced techniques employed by CRRU-MADRL. In difference, the substantial improvements in energy efficiency achieved by CRRU-MADRL are attributed to their advanced task scheduling strategies that optimize the heterogeneous computational core-based systems, which integrates both smaller-core and larger-core implementation models. These strategies, as outlined in (20) and (23), enable better load-performance distribution, enhanced task scheduling, and alignment with performance constraints. Consequently, energy consumption is significantly reduced, as demonstrated in (20)-(23). The improvements in energy efficiency, evident in Figures 2 and 3, underscore the effectiveness of CRRU-MADRL in outperforming REWS and MOWS in terms of execution time and scheduling efficiency.

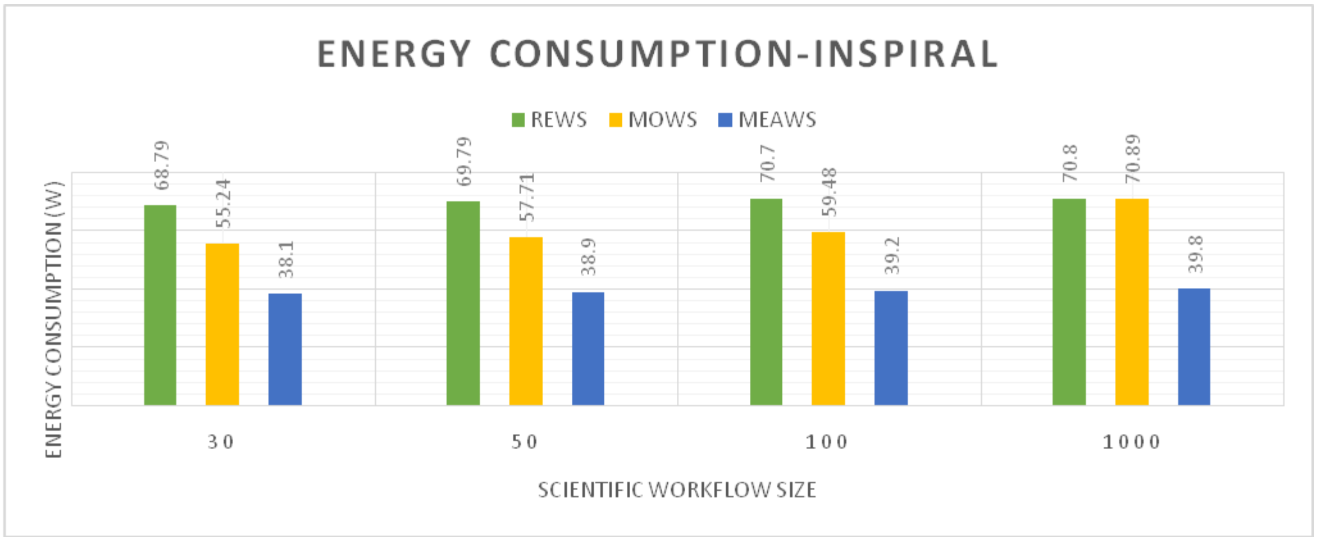


Figure 2. Energy consumption vs Inspiral workflow size.

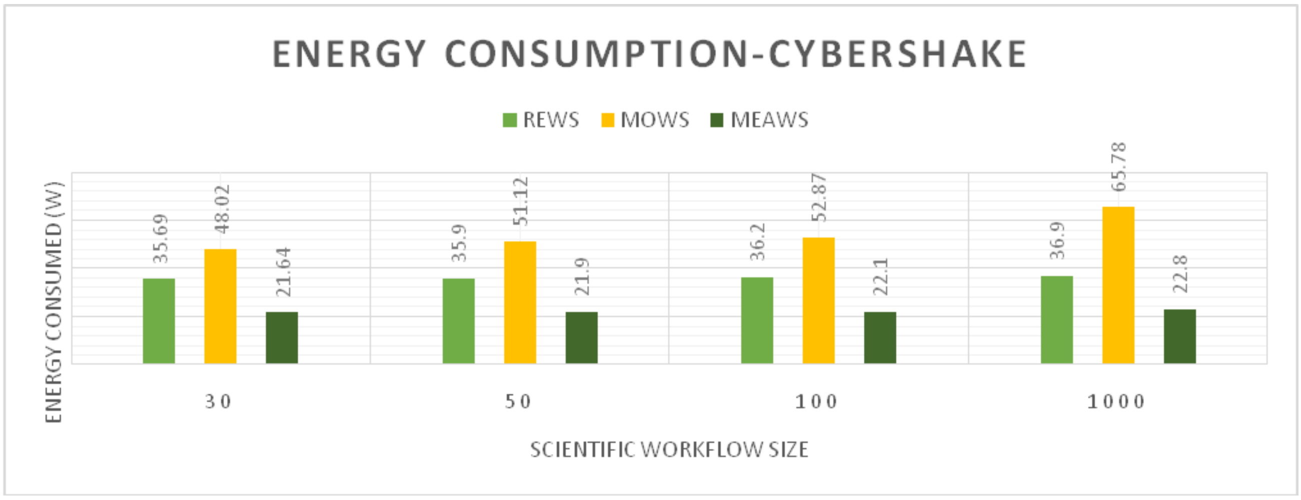


Figure 3. Energy consumption vs CyberShake workflow size.

Makespan Performance:

This section assesses the makespan efficiency of three scheduling frameworks namely REWS, MOWS, and CRRU-MADRL by analyzing the makespan needed to finish execution of scientific workflows of different sizes. The makespan, defined as the total time to execute a given set of tasks, is a critical performance metric in scientific computing. Figure 4 demonstrates the makespan performance when executing the Inspiral workflow across varying task sizes. The results display that CRRU-MADRL significantly reduces the makespan by 81.8% and 76.7% when compared to REWS and MOWS, respectively. The results indicate that CRRU-MADRL outperform the baseline models by a considerable margin, making them highly efficient for executing large and complex workflows. Figure 5 exhibits the makespan outcome for Cybershake workflows, which include diverse size with varying computational demands. The results reveal that CRRU-MADRL reduces the makespan by 95.7% and 64.41% compared to REWS and MOWS, respectively. These improvements highlight CRRU-MADRL as superior frameworks in minimizing makespan while processing complex scientific workflows. The extended makespan reflected in REWS and MOWS can be credited to the inherent restrictions of each method. EMS primarily focuses on task allocation to VCNs with superior consistency and lower energy spending but does not rank optimizing makespan, resulting in longer execution times. MOWS, on the other hand, utilizes soft computing models for scheduling tasks but is not specifically tailored for heterogeneous platforms, which leads to suboptimal task scheduling when compared to the advanced techniques employed by CRRU-MADRL. In difference, the substantial improvements in makespan performance achieved by CRRU-MADRL are attributed to their advanced task scheduling strategies that optimize the heterogeneous computational core-based systems, which integrates both smaller-core and larger-core implementation models. These strategies, as outlined in (20) and (23), enable better load-performance distribution, enhanced task scheduling, and alignment with performance constraints. Consequently, the execution time is significantly reduced, as demonstrated in (20)-(23). The improvements in makespan performance, evident in Figures 4 and 5, underscore the effectiveness of CRRU-MADRL in outperforming REWS and MOWS in terms of execution time and scheduling efficiency.

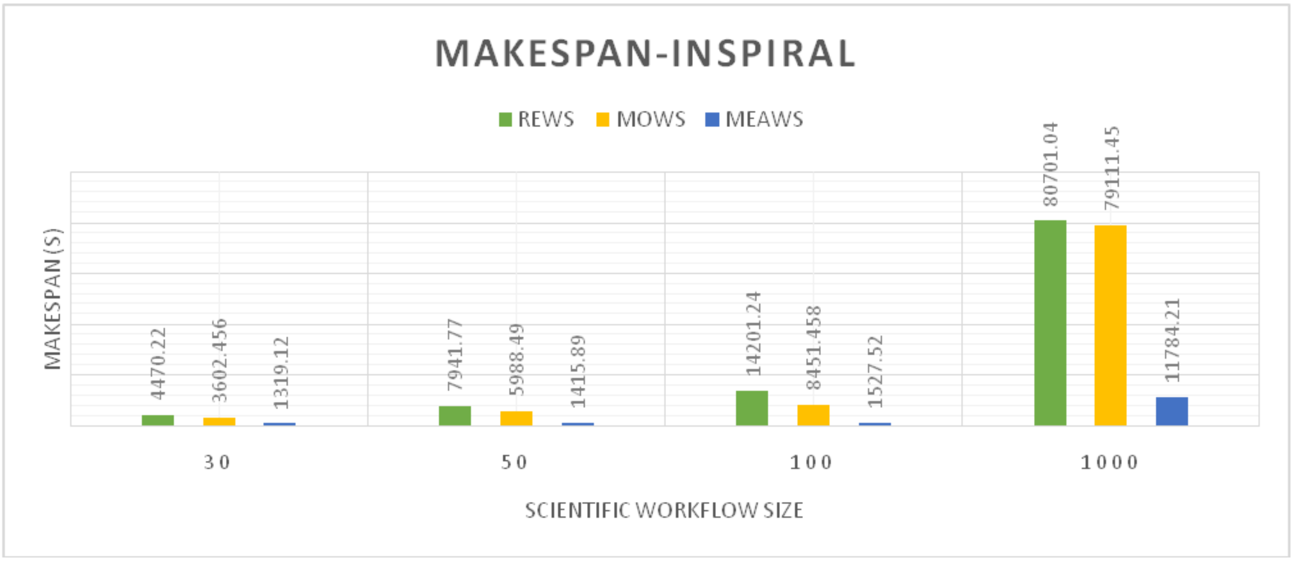


Figure 4. Makespan vs Inspiral workflow size.

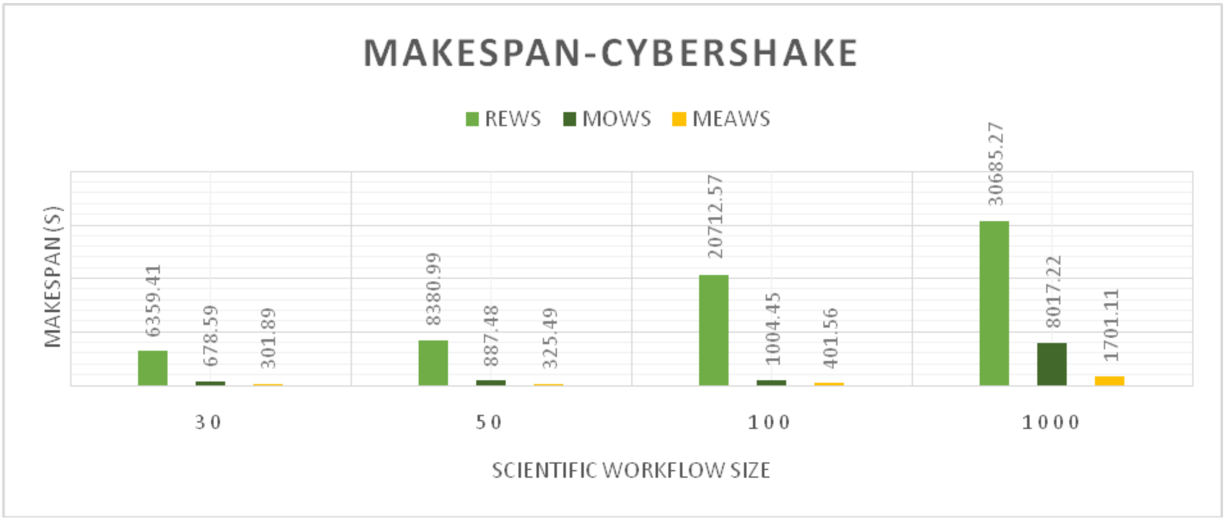


Figure 5. Makespan vs CyberShake workflow size.

Total Cost Performance:

This section evaluates the cost efficiency of three workflow scheduling models REWS, MOWS, and CRRU-MADRL based on total cost incurred for executing scientific workflows of varying sizes. Total cost here includes computational resources consumed over time and energy, a key metric for large-scale cloud and edge-cloud infrastructures. Figure 6 illustrates the total cost incurred for executing the Inspiral workflow with different task sizes (30, 50, 100, 1000). The proposed CRRU-MADRL framework demonstrates significant cost savings: At 30 tasks, CRRU-MADRL reduces total cost by 70.37% compared to REWS, 63.24% compared to MOWS, At 50 tasks, CRRU-MADRL reduces total cost by, 83.29% compared to REWS, 77.84% compared to MOWS, At 100 tasks, CRRU-MADRL reduces total cost by, 89.96% compared to REWS, 83.14% compared to MOWS, At 1000 tasks, CRRU-MADRL achieves a drastic cost reduction of 98.09% compared to REWS, 98.05% compared to MOWS. These improvements are attributed to CRRU-MADRL's dynamic core-aware task allocation and energy-aware reinforcement scheduling which collectively enhance resource reutilization across heterogeneous environments. Figure 5 presents the total cost for CyberShake workflow across increasing task complexities: At 30 tasks, CRRU-MADRL achieves cost reduction of, 95.16% compared to REWS, 55.49% compared to MOWS, At 50 tasks, CRRU-MADRL achieves cost reduction of, 96.04% compared to REWS, 63.21% compared to MOWS, At 100 tasks, CRRU-MADRL reduces cost by, 98.03% compared to REWS, 59.97% compared to MOWS, At 1000 tasks, the model shows a reduction of, 94.43% compared to REWS, 78.75% compared to MOWS. These results confirm that CRRU-MADRL offers scalable and energy-conscious scheduling, especially beneficial for high-load scientific workflows deployed in edge-cloud environments. The consistent superiority of CRRU-MADRL over both REWS and MOWS across all task sizes and workflow types underlines its effectiveness in minimizing execution costs. These results reinforce the framework's capability to maintain high resource utilization and execution efficiency while adhering to energy and cost constraints, positioning it as a practical solution for sustainable scientific workflow scheduling.

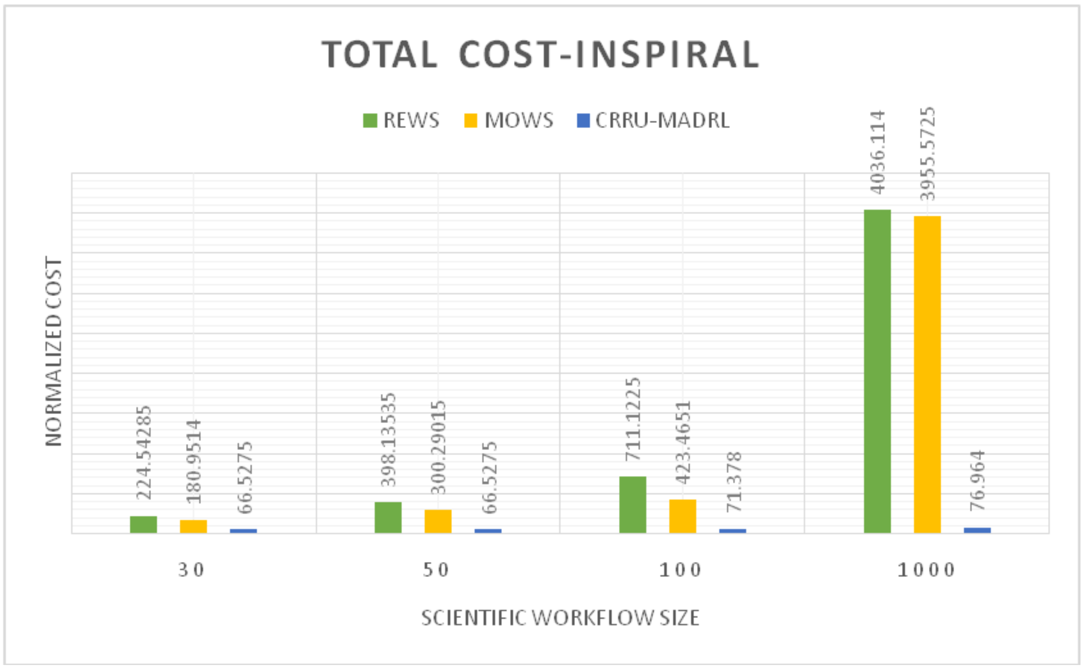


Figure 6. Cost vs Inspiral workflow size.

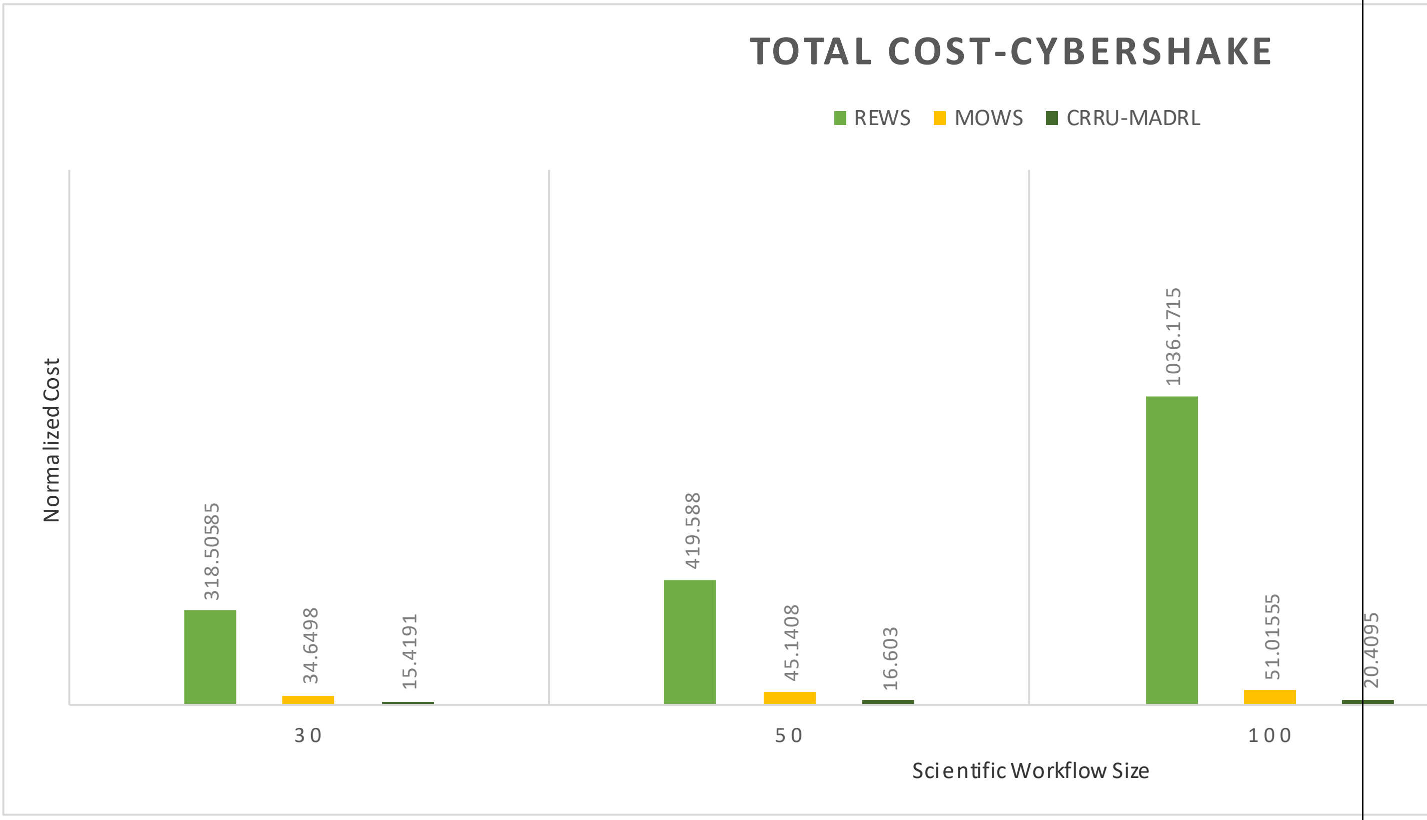


Figure 7. Cost vs CyberShake workflow size.

Future work to be carried out

- To design quality and cost-effective metrics for virtual resource function optimization for scientific workflow provisioning in edge-cloud platform.
- To develop service function chain reutilization optimization for meeting scientific workflow task multi-level deadline constraint.
- To design and develop cost-reliability metrics for reusability of virtual resource function and service chaining optimization for execution complex multiple scientific workflows in edge-cloud platform.
- Publication in good Scopus and Web-of-Science indexed journal
- Development of objective 2 to 4 in Cloudsim simulator will be done.
- Collection of different scientific work benchmarks will be done.
- Mathematical model and paper writeup of objective 2 to 4 will be carried out.

REFERENCES

[1] Erbel and J. Erbel, J., Grabowski, J. Scientific workflow execution in the cloud using a dynamic runtime model. Softw Syst Model 23, 163–193 (2024). https://doi.org/10.1007/s10270-023-01112-6.
[2] Tewari, A., Goyal, N., Awasthi, L. K., & Priyanka. (2025). Efficient Workflow Scheduling in Cloud Computing Using Hybrid Algorithm. IETE Journal of Research, 1–12. https://doi.org/10.1080/03772063.2024.2448263board
[3] Hind Mikram, Said El Kafhali, Youssef Saadi, HEPGA: A new effective hybrid algorithm for scientific workflow scheduling in cloud computing environment, Simulation Modelling Practice and Theory, Volume 130, 2024, 102864, ISSN 1569-190X, https://doi.org/10.1016/j.simpat.2023.102864.
[4] Zhang, X. Optimizing scientific workflow scheduling in cloud computing: a multi-level approach using whale optimization algorithm. J. Eng. Appl. Sci. 71, 175 (2024). https://doi.org/10.1186/s44147-024-00512-9

[5] Saeed, H.A.; Al-Janabi, S.T.F.; Yassen, E.T.; Aldhaibani, O.A. Survey on Secure Scientific Workflow Scheduling in Cloud Environments. Future Internet 2025, 17, 51. https://doi.org/10.3390/fi17020051
[6] Uma K.M., Shailendra Shukla, Energy and performance-aware workflow scheduler using dynamic virtual network resource optimization under edge-cloud platform, Computers and Electrical Engineering, Volume 123, Part B, 2025, 110085, https://doi.org/10.1016/j.compeleceng.2025.110085.
[7] Xingjuan Cai, Yan Zhang, Mengxia Li, Linjie Wu, Wensheng Zhang, Jinjun Chen, Dynamic deadline constrained multi-objective workflow scheduling in multi-cloud environments, Expert Systems with Applications, Volume 258, 2024, 125168, https://doi.org/10.1016/j.eswa.2024.125168.
[8] Alsadie, D., Alsulami, M. Enhancing workflow efficiency with a modified Firefly Algorithm for hybrid cloud edge environments. Sci Rep 14, 24675 (2024). https://doi.org/10.1038/s41598-024-75859-3
[9] Rateb, R., Hadi, A.A., Tamanampudi, V.M. et al. An optimal workflow scheduling in IoT-fog-cloud system for minimizing time and energy. Sci Rep 15, 3607 (2025). https://doi.org/10.1038/s41598-025-86814-1.
[10] Andriulo, F.C.; Fiore, M.; Mongiello, M.; Traversa, E.; Zizzo, V. Edge Computing and Cloud Computing for Internet of Things: A Review. Informatics 2024, 11, 71. https://doi.org/10.3390/informatics11040071
[11] K Karim F, Ghorashi S, Alkhalaf S, H A Hamza S, Ben Ishak A, Abdel-Khalek S. Optimizing makespan and resource utilization in cloud computing environment via evolutionary scheduling approach. PLoS One. 2024 Nov 22;19(11):e0311814. doi: 10.1371/journal.pone.0311814.
[12] A. Tarraf et al., "Malleability in Modern HPC Systems: Current Experiences, Challenges, and Future Opportunities," in IEEE Transactions on Parallel and Distributed Systems, vol. 35, no. 9, pp. 1551-1564, Sept. 2024, doi: 10.1109/TPDS.2024.3406764. SURVEY
[13] A. Vivas, A. Tchernykh and H. Castro, "Trends, Approaches, and Gaps in Scientific Workflow Scheduling: A Systematic Review," in IEEE Access, vol. 12, pp. 182203-182231, 2024, doi: 10.1109/ACCESS.2024.3509218. SURVEY
[14] Jinchao Chen, Pengcheng Han, Ying Zhang, Tao You, Pengyi Zheng, Scheduling energy consumption-constrained workflows in heterogeneous multi-processor embedded systems, Journal of Systems Architecture, Volume 142, 2023, 102938, https://doi.org/10.1016/j.sysarc.2023.102938.
[15] S. Mangalampalli et al., "Multi Objective Prioritized Workflow Scheduling Using Deep Reinforcement Based Learning in Cloud Computing," in IEEE Access, vol. 12, pp. 5373-5392, 2024, doi: 10.1109/ACCESS.2024.3350741.
[16] M. Fan, X. Zhao, X. Zuo and L. Ye, "A Budget-Constrained Workflow Scheduling Approach With Priority Adjustment and Critical Task Optimizing in Clouds," in IEEE Transactions on Automation Science and Engineering, 2025. doi: 10.1109/TASE.2024.3456762.
[17] S. Tao, Y. Xia, L. Ye, C. Yan and R. Gao, "DB-ACO: A Deadline-Budget Constrained Ant Colony Optimization for Workflow Scheduling in Clouds," in IEEE Transactions on Automation Science and Engineering, vol. 21, no. 2, pp. 1564-1579, 2024, doi: 10.1109/TASE.2023.3247973.
[18] S. -E. Chafi, Y. Balboul, M. Fattah, S. Mazer and M. El Bekkali, "Novel PSO-Based Algorithm for Workflow Time and Energy Optimization in a Heterogeneous Fog Computing Environment," in IEEE Access, vol. 12, pp. 41517-41530, 2024, doi: 10.1109/ACCESS.2024.3377236.
[19] J. Perez-Valero, A. Banchs, P. Serrano, J. Ortín, J. Garcia-Reinoso and X. Costa-Pérez, "Energy-Aware Adaptive Scaling of Server Farms for NFV With Reliability Requirements," in IEEE Transactions on Mobile Computing, vol. 23, no. 5, pp. 4273-4284, May 2024, doi: 10.1109/TMC.2023.3288604.
[20] Ijaz, S., Ahmad, S.G., Ayyub, K. et al. Energy-efficient time and cost constraint scheduling algorithm using improved multi-objective differential evolution in fog computing. J Supercomput 81, 116, 2025. https://doi.org/10.1007/s11227-024-06550-7.
[21] X. Wang, J. Cao and R. Buyya, "Adaptive Cloud Bundle Provisioning and Multi-Workflow Scheduling via Coalition Reinforcement Learning," in IEEE Transactions on Computers, vol. 72, no. 4, pp. 1041-1054, 2023, doi: 10.1109/TC.2022.3191733.
[22] H. Lahza, B. R. Sreenivasa, H. F. M. Lahza, and Shreyas J, “Adaptive Multi-Objective Resource Allocation for Edge-Cloud Workflow Optimization Using Deep Reinforcement Learning,” Modelling, vol. 5, no. 3, pp. 1298-1313, 2024. https://doi.org/10.3390/modelling5030067.
[23] Krishna MSR and Mangalampalli S. PWSA3C: Prioritized Workflow Scheduler in Cloud Computing Using Asynchronous Advantage Actor Critic (A3C) Algorithm. in IEEE Access, vol. 12, pp. 127976-127992, 2024, https://10.1109/ACCESS.2024.3457518.
[24] S. Sudheer Mangalampalli, G. Reddy Karri, P. Reddy Ch, K. Sree Pokkuluri, P. Chakrabarti and T. Chakrabarti, "An Energy and Temperature Aware Deep Reinforcement Learning Workflow Scheduler in Cloud Computing," in IEEE Access, vol. 12, pp. 163424-163443, 2024, doi: 10.1109/ACCESS.2024.3488965.

[25] L. Zhang, M. Ai, K. Liu, J. Chen and K. Li, "Reliability Enhancement Strategies for Workflow Scheduling Under Energy Consumption Constraints in Clouds" in IEEE Transactions on Sustainable Computing, vol. 9, no. 02, pp. 155-169, 2025. doi: 10.1109/TSUSC.2023.3314759.
[26] Son, Jungmin & He, Tianzhang & Buyya, Rajkumar. (2019). CloudSimSDN-NFV: Modeling and simulation of network function virtualization and service function chaining in edge computing environments. Software: Practice and Experience. 49. 10.1002/spe.2755.
[27] Siavashi, A., Momtazpour, M. GPUCloudSim: an extension of CloudSim for modeling and simulation of GPUs in cloud data centers. J Supercomput 75, 2535–2561, 2019. https://doi.org/10.1007/s11227-018-2636-7.
[28] Chen W and Deelman E. WorkflowSim: A toolkit for simulating scientific workflows in distributed environments. in Proc. IEEE 8th Int. Conf. E-Sci., 2012, pp. 1–8. https://doi.org/10.1109/eScience.2012.6404430.
[29] Juve G, Chervenak A, Deelman E, Bharathi S, Mehta G, and Vahi K. Characterizing and profiling scientific workflows. Future Generation Computer Systems, vol. 29, no. 3, pp. 682–692, 2013. https://doi.org/10.1016/j.future.2012.08.015.